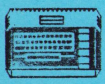
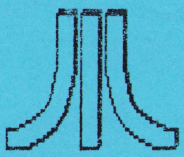
T — Y — N — E — & — W — E — A — R



TYNE & WEAR



ATARI [8]-BIT USER GROUP

Newsletter of TWAUG

Software
Editorial
Buy & Sell
Hardware

Reviews
Help line
Section
Repair Info

Public Domain Library

ISSUE #8

March/April 1994

U — S — E — R — G — R — O — U — P

BRING YOUR EIGHT UP TO DATE with power products from COMPUTER SOFTWARE SERVICES

THE BLACK BOX

The BLACK BOX is an add-on board for the Atari 600XL,800XL and 130XE 8-bit computers. It is a T-shaped board that plugs into the PBI port of the XL computer, or the ECI and cartridge ports of the 130XE. Connectors for both types of computers are built into the BLACK BOX so no adapter boards are necessary. A cartridge port is available on the board itself for 130XE users.

The BLACK BOX provides many unique and useful functions. The four primary functions are:-

* RS-232 serial modem port
* Parallel printer port
* SASI/SCSI hard disk port
* Operating System enhancements

The BLACK BOX is $199.95 for the basic unit, and $249.95 with an onboard 64K printer buffer.
Shipping and Handling extra.

THE BLACK BOX ENHANCER

A must for all BLACK BOX owners. The BLACK BOX ENHANCER is a plug-in module for your BLACK BOX, enhancing the printer functions and adding an instantly available, full featured sector editor!

Installation of the BLACK BOX ENHANCER requires one simple solder connection. Only $49.95 plus shipping/handling.

THE FLOPPY BOARD

Our latest and greatest product. The FLOPPY BOARD is an add-on expansion board for the BLACK BOX interface. It allows the use of the same inexpensive floppy drive mechanisms used in IBM computers. The FLOPPY BOARD is the first floppy drive interface to support "high density" floppy drive mechanisms in either 5.25 inch or 3.5 inch. Built into the FLOPPY BOARD are our BLACK BOX ENHANCER and a version of our SUPER ARCHIVER to allow copying of protected disks for 3.5 inch format. Included with the FLOPPY BOARD is our program to read and write to IBM or ST formatted disks. This makes the FLOPPY BOARD the best way to transfer files to and from your 8-bit.

The FLOPPY BOARD is only $149.95 plus shipping & handling.

THE MULTIPLEXER

This device brings the power and flexibility of larger systems to your 8-bit. The Multiplexer is a collection of cartridge interface boards that allow up to 8 Ataris to read and write to the same drives (typically a hard disk), access the same printer(s), and talk to each other. It is the first practical networking system for the Atari 8-bit computer.

One "master" computer (any 8-bit) is equipped with the master Multiplexer interface. Then up to 8 "slave" computers hook up to this master, each having their own slave interface.

The "common" peripherals (things that are to be shared) are connected to the master. On each slave, all disk and printer I/O is routed through the master, so no extra disk drives are needed.

The Multiplexer sells for $199.95 for a master and two slave units with cable. Additional slave units are $89.95 each, plus shipping/handling.

THE SUPER ARCHIVER II

The SUPER ARCHIVER II edits and copies all enhanced density programs plus retains all the features of the SUPER ARCHIVER.

The SUPER ARCHIVER II is only $99.95 plus shipping $ handling. NOTICE: if you already have THE SUPER ARCHIVER you may upgrade to S.A.II for only $29.95 plus shipping/handling. Software only.

THE BIT WRITER

The Super Archiver BIT WRITER is capable of duplicating even the "uncopyable" Electronic Arts and Synapse Syn-series, which employ 34 full sector tracks. The BIT WRITER must be used with the SUPER ARCHIVER

The BIT WRITER is only $79.95 plus shipping/handling.

THE ULTRA SPEED PLUS OS

The Operating System that should be in every XL/XE computer! The Ultra Speed Plus puts unbelievable speed and convenience at your fingertips.

Use any DOS to place Ultra Speed formats on your disks (with XF551 or modified 1050 drives), reading and writing at this speed with most programs. This high speed mode can be turned off for maximum compatibility.

Four simple solder connections are required for installation if your machine has a socketed OS ROM. The Ultra Speed OS is only $69.95 plus shipping/handling.

For more information on these and other 8-bit products:

CONTACT
COMPUTER SOFTWARE SERVICES
PO BOX 17660
ROCHESTER, NEW YORK 14617
USA

ORDERING LINE: (716) 429-5639
FAX: (716) 247-7158
BBS: (716) 247-7157

or contact T.W.A.U.G. we will do our best to help.

EDITORIAL

Who to blame!!!

John Matthewson
David Ewens
Max Gerum

EXCHANGE RATES!

I've had a request to publish the international money exchange rate. I wouldn't mind publishing the rate every two month, but would it be any good?

The reason for saying this is, I've purchased the Black Box from the U.S.A. and when I received the invoice the rate was entirely different than at the time of purchase. The exchange rate changes two, three or more times a day, sometimes fluctuates umpteen times an hour some days.

The daily rate between the Dollar and the english Pound usually moves up or down between $1.42 and $1.58. Calculate between these rates and you are never far off the actual rate.

SPRING ALL MICRO SHOW.

There is only one more month to go before the Spring show, it takes place on Saturday the 16th of April.

TWAUG will again have a stand there and one extra person from the team, who was missing at the November show due to an operation, will also attend this time. I must say I am really looking forward to it.

Every body please make an efford to attend this show, you never know you could pick up a bargain.

The next issue will be ready by mid-May.

CONTENTS

EDITORIAL 3

MICRO DISCOUNT
Full page advert 4

HINTS & TIPS
& COMPUTER TYPIST DEMO 5

BASIC KEYPRESS RESPONSE
by Ralph Bradley &
TRANSDISK IV REVIEW
by Mark Stinson 6

MARK's GAMES COLUMN
more Adventures reviewed
by Mark Stinson 8

GAMES REVIEWS
by Mark Fenwick 9

PARALLEL BUS REVEALED
PART III by Earl Rice 10

TEXTPRO WORD PROCESSOR
by M.Gerum 18

DISK CONTENT 21

PUBLIC DOMAIN REVIEW
by Bill Jackson 22

GOOD NEWS
news about the new book 25

DD3 & PAGE 6 WRITER
corrections to last issues article
by John Bunting
JUST A BYTE OF HUMOR
re-print from OL'HACKERS newsletter
NEWS from GERMANY by Robby 26

CRACKING THE CODE
by Keith Mayhew 27

BRITISH SPARTADOS TIME
by Terry Chamberlain 32

OL'HACKERS USER GROUP PHOTOS
a few lines by David Ewens
BUY & SALE 34

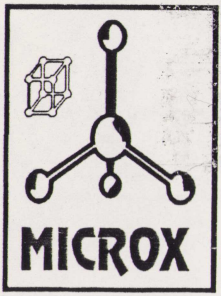
ATARI CLASSICS, OL'HACKERS
ATARI USER GROUP & ULTIFONT
advertisement full page 35

RICHARD GORE, PHOENIX, PAGE 6
& MICRO DISCUNT
Advertisements for support and new
releases. 36

Are You Still Complaining about lack of New software for your Atari 8 bit ?
If you are you don't know about Mail Order From

MICRO-DISCOUNT

265, CHESTER ROAD.
STREETLY.
WEST MIDLANDS.
B74 3EA.
ENGLAND.



TEL 021 353 5730
FAX 021 352 1669

ADAX *HANS KLOSS*

DARKNESS HOUR

HINTS AND TIPS.

The items in this issue of hints and tips are a collection of small programs that I hope will be both interesting and useful to those who still like to do a little programming in Atari Basic.
I would have liked to say that some of the items had been sent in by some of you out there but unfortunately not. I'm sure that someone out there must have some useful bits of info they would like to share with other Atarians, why not get your thinking caps on and send something in for future issues or, I may have to drop this column soon.

Listing 1 is called Joystick Reader. It can return 18 possible values instead of the usual 9. This machine language routine would most likely be used for games. It uses the format X=USR(ADDR,STKNUM). ADDR is the address of the routine and STICKNUM is the joystick port to be read. Variable X returns the normal joystick directional value that you would expect when the joystick button is not pressed, but it adds 128 to the value if the button is pressed.

```
10 DIM A$(32):STIKNUM=0
20 FOR I=1 TO 32:READ X:A$(I)=CHR$(X):NEXT I
30 X=USR(ADR(A$),STKNUM):? X:GOTO 30
50 DATA 104,104,104,170,189,120,2,188,132,2,208,5,160,128,184
,80,2,160,0,132,203,24,216,101,203,160,0,132
60 DATA 213,133,212,96
```

Listing 2 is a quick and easy way to generate the famous Atari Rainbow effect. RUN the program and the rainbow starts. Exit by pressing the (START) key. Call this routine with Z=USR(ADDR,X). ADDR is the address again, and X is the number of the colour register affected. Subtract 704 from the colour register location to find the value of X. This will work for Player/Missile graphics as well.

```
10 CT=0:X=8:REM BACKGROUND
20 CT=CT+1:READ Q:IF Q=-1 THEN 40
30 POKE 1663+CT,Q:GOTO 20
40 Z=USR(1664,X)
50 END
60 DATA 104,104,104,168,232,142,10,212,138,153,18,208,169,6,20
5,31,208,208,241,96,-1
```

BIGFOOT

The techniques in Bigfoot could be used in larger programs to produce some fantastic effects. I like the way this program makes transitions from one effect to another, rather than just repeating itself. And the clever use of sound added to the overall impact of the program.

```
1 GRAPHICS 10:FOR X=0 TO 15 STEP +2 POKE 704+(X/2),X:NEXT
X
2 SOUND 0,200,10,3:SOUND 1,202,10,3:UPPER=PEEK(560)+256*PE
EK(561)+4:LOWER=PEEK(560)+256*PEEK(561)+100
3 FOR X=0 TO 79:C=C*(C(8)+1:COLOR C:PLOT X,0:DRAWTO X,178:
NEXT X:SOUND 0,190,10,4:SOUND 1,188,10,4
4 FOR Y=1 TO 175 STEP +10:FOR X=66 TO 12 STEP -1:C=C*
(CC(8)+1:COLOR C:PLOTX,Y:DRAWTO X,Y+5:NEXT X:NEXT Y
5 FOR Y=1 TO 15:FOR X=1 TO 40:POKE UPPER,X:SOUND 0,200
+X,10,Y:NEXT X:FOR X=1 TO 40:POKE LOWER,X:NEXT X:NEXT Y
6 SOUND 1,200+X+2,10,13:SOUND 2,200+X+4,10,14:SOUND 0,0,0,0
7 FOR Y=1 TO 4:FOR X=1 TO 40:POKE UPPER,X:GOSUB 9:NEXT X:
FOR X=1 TO 40:POKE LOWER,X:GOSUB 9:NEXT X:NEXT Y
8 FOR X=0 TO 15 STEP 0.1:POKE 764+(X/2),0:SOUND 1,200+X+2
,10,15-X:SOUND 2,200+X+4,10,15-X:NEXT X:END
9 Z=PEEK(712):POKE 712,PEEK(711):POKE 711,-EEK(710):POKE 710,
PEEK(709):POKE 709,PEEK(708):POKE 708,PEEK(707)
10 POKE 707,PEEK(706):POKE 706,PEEK(705):POKE 705,2:RETURN
```

SUNSET

When you RUN this one, be sure to adjust your colour so that the uppermost part of the screen is sky blue.

```
1 GRAPHICS 11:C=5:FOR V=1 TO 136:A=136:B=191:D=78
2 C=C+0.07:COLOR C:PLOT 0,V:DRAWTO 78,V:NEXT V:X=0:Y=136
3 COLOR 0:R=INT(RND(0)*30)+1:FOR V=1 TO R:PLOT X,136:DRAWTO
X,V:X=X+1:Y=Y-1:IF X)=78 THEN 7
4 NEXT V
5 FOR V=1 TO R:PLOT X,136:DRAWTO X,Y:X=X+1:Y=Y+1:IF X)=78
THEN 7
6 NEXT V:GOTO 3
7 FOR N=1 TO 20:COLOR 7:PLOT 1,N:DRAWTO D,N:NEXT N:COLOR
11:FOR Y=A TO B:PLOT 0,Y:DRAWTO D,Y:NEXT Y:V=10:UD=UD+1
8 COLOR 0:PLOT 60,V:DRAWTO 66,15:DRAWTO 72,V:COLOR 7:PLOT
60,V:DRAWTO 66,15:DRAWTO 72,V:V=V+UD:IF V=20 THEN UD=-1
9 IF V=10 THEN UD=+1
10 GOTO 8
```

COMPUTER TYPIST DEMO

This short but interesting subroutine shows how to print text on screen character-by-character with a typewriter sound. This is the same effect you've probably seen in programs like HACKER.

When you RUN the program, instructions will write themselves out onto the screen. You can put up to 40 characters at a time in to the string A$. Then GOSUB 1000 to print the characters on screen, and move on to the text in the next A$.

If you want a different sound, just experiment with the SOUND statement in line 1030.

Computer Typist Demo Listing on next page:

HINTS & TIPS continued

COMPUTER TYPIST LISTING

```
10 ? "K":DIM A$(40)
20 REM ADD LINES TO BE PRINTED HERE
30 A$="This is an example of computer
TYPE":GOSUB 1000
40 A$="as seen in some games and progr
ams.":GOSUB 1000
50 PRINT
60 A$="Type LIST to see this program."
:GOSUB 1000:A$="Delete lines 30-90 and
 add your own":GOSUB 1000
65 A$="text lines.":GOSUB 1000
70 PRINT
80 A$="Put your text (1-40 characters)
in":GOSUB 1000
90 A$="A$, and GOSUB 1000.":GOSUB 1000
99 PRINT :END
1000 REM TYPE SUBROUTINE
1010 PRINT :FOR Q=1 TO LEN(A$)
1020 IF A$(Q,Q)=" " THEN FOR W=1 TO 10
:NEXT W:? " ";:GOTO 1040
1030 FOR W=20 TO 0 STEP -5:SOUND 0,W,4
,6:NEXT W:? A$(Q,Q);:SOUND 0,0,0,0
1040 NEXT Q
1050 RETURN
```

KEY2.BAS

by Ralph Bradley

This Programme will test your Keyboard Response Time.

EXAMPLE

RUN PROGRAMME
' FIND: A ' displayed
Press Letter Key 'A'
RESPONSE Time given

After a few seconds the next Character to FIND will be Displayed.

Press RESET to exit programme.

The program listing is on Side A of this issue disk.

TRANSDISK IV

by Mark Stinson

I am sure that every reader of New Atari User will have seen, used, or heard about Transdisk IV, the tape to disk utility. I am equally sure that a large number of disk drive owners will not have purchased Transdisk because of their reluctance to accept that a program so reasonably priced could possibly do all that it claims.

TRANSDISK IV continued

In this article I will try show some examples of the uses, methods and limitations of Transdisk, as well as giving some hints to overcome the minor problems I have found. I would hope that if the 8-bit, drive owning community were aware of this program's capabilities, they would be prepared to part with twelve pounds ninety five pence in order to improve their own system, and at the same time help keep our magazine alive.

INTRODUCTION

So what, you may ask, does Transdisk actually do? Well, Transdisk allows you to create a disk loader menu, and to copy commercial and non commercial cassette software to that disk. You need not worry about breaching copyright, as long as you only make backup copies of your own software, and solely for your own personal use.

Transdisk is capable of copying most cassettes, including software that requires 64K of memory, and games that load extra levels. As there are a large number of different cassette formats which can make copying tricky, the program has an equally impressive number of variable settings to deal with non standard recordings. However, I have found from personal experience that the most difficult cassettes to copy are those which load extra levels, some of which I have still not cracked. As I mentioned earlier, Transdisk can handle these cassettes , but they can be difficult. One specific problem I have encountered in this area has been the limited storage capacity of a disk when extra levels have to be stored.

USES

Most people who own a disk drive will most probably have started out with a cassette deck, and cassette based software. I know that when I first bought my drive, some five years ago, I had upwards of 70 cassettes. All but five of these have been successfully transferred to disk, and now load in seconds rather than minutes.

Even if you do not own a single cassette, Transdisk is still well worth the investment. The market for Atari 8-bit software is dwindling, and along with it goes the availability of disks. But even in this sorry state of the market, cassettes are still being released. Just take a look at the inside front cover of New Atari User, issue 56; do you own all those games? I would doubt that many people own them all, and I am sure many people don't bother because they are stored on cassette. Once you own Transdisk, the availability of software for your Atari grows ever wider.

Transdisk is also good news for 130XE or expanded memory owners. This was one of the few packages written to use the extra memory available on a 130XE, while remaining compatible with smaller memory machines. It has also been around for a number of years, so there is no shortage of users to communicate with, and gain assistance from.

TRANSDISK IV continued

TRANSDISK IN ACTION

When you boot up the disk, you are presented with the following menu:

1 Index of drive one files
2 Index of drive two files
Initialize disk
Standard autoboot tape read
Non-Standard autoboot tape read
Write new disk file
Append to disk file
Delete disk file

90% of the time, using Transdisk is quick and easy. Having a 130XE, or expanded XL also help to limit the time you need to invest in copying your cassettes. So, with the exception of the odd few problematic cassette formats, the following stages are all that need to be undertaken to copy cassettes, and place them on disk.

(I)nitialize disk

This option enables the user to format a disk for either 64K or standard games. At the sub-menu select either XLMENU.SYS (XEMENU.SYS on the 130XE) for 64K games, or XTMENU.SYS for standard games. Your blank disk is then formatted, and the relevant menu driver written to it.

(S)tandard autoboot tape read

This is the option used for transferring most cassettes. With a tape inserted, the cassette deck is set to 'play', and the 'start' key is pressed. The software will now read in the first stage of the cassette; some cassettes are only one stage long, but most are made up of several. If your cassette is made up of several stages this will be apparent during a normal cassette load. When loading, the recorded hisses and crackles will be interrupted with the occasional gap, normally lasting two or three seconds. If you know the number of stages a cassette is made up of (load the tape normally, and count them) then 'option/select' can be used to set the number stages needed. This is no problem on a 130XE, but you are limited to the number of stages a smaller memory machine can load at once.

(W)rite disk file

This option writes the game file, or stage, to any menu disk. You may write the file out as many times as you wish.

Now, booting the new disk will give a menu showing the games currently stored on it. Simply press the letter which corresponds to the required game/program and if all is well, the game should load.

If you own a 48K or 64K machine there will be many games which cannot be held in memory in full (the Transdisk program uses a fairly hefty chunk of memory). This is especially noticeable with games which load extra levels. This is where the 'append' function comes into play.

The cassette is copied in the usual manner, but with the number of stages set to 1. The one stage copy is then loaded and written to disk. Each subsequent stage is loaded, and 'appended' to the main file until the game is copied. The only drawback is the length of time the user has to spend at the keyboard, waiting for the individual stages to load.

NON-STANDARD CASSETTES

Some cassettes will not copy with the standard settings, and the non-standard option will have to be used. This is well covered in the instructions, but briefly, it is used as follows. Different tapes may have different 'baud' settings. A baud setting relates to the speed and size of portions of data being transmitted to the computer, via the tape deck. The non-standard option allows you to choose different settings for baud rate- 0,2,4,6 and 8. Anyone who owns Fighter Pilot will have noticed the different sound that the cassette generates when loading; this will not copy on a standard setting. However, by setting the baud rate to eight, Transdisk will copy Fighter Pilot with ease.

IN CONCLUSION

Transdisk is a very powerful utility and is a must for all disk owners. It does all that it claims, and usually with little fuss, and the minimum of effort. It does, however, have some limitations:

1. COMPATIBILITY

Although it is compatible with DOS 2, it is not easy to backup menu disks. The best I have managed is to format a disk with Transdisk and then delete the menu. Next, load DOS 2 and use 'duplicate disk'. This will copy a whole menu disk, but I usually find that one of the files will not work properly. To combat this, I then use transdisk to copy a stage from any cassette and write a dummy file to the menu disk. Once the menu disk is copied, the dummy file is usually the file corrupted.

2. DISK SPACE

The menu file takes a large number of sectors (120) so games like Gauntlet will not fit on a disk.

3. GAMES WITH EXTRA LEVELS

Games that load extra levels are especially difficult to copy. I have still not managed to copy Gauntlet and The Goonies.

4. GAME SAVES

Cassette games which use game save (such as adventures) do not have the save facility ported to disk. However, I have found that this can be overcome with many adventures such as Mordon's Quest, Adventure Quest, and all the Scott Adams tapes. Simply press system reset when the game is loaded. Sometimes this re-boots the game, in which case it will not work. However, if the game reappears instantly, you can now save to cassette.

TRANSDISK IV continued

This is not particularly annoying, as most game save facilities only take a few seconds anyway.

One final hint that may be useful is that DOS 2 can read the disk. So, you can use a sector editor to examine the copied software, and you can use a DOS 2 based cataloguer.

Despite the limitations, I have absolutely no hesitation in recommending this utility. Indeed, I would suggest that this is the one essential disk in all drive owners' collections.

Availability: Page 6 Publishing

Price: twelve pounds ninety five pence

MARK's GAMES COLUMN

DALLAS QUEST & HACKER solutions

by Mark Stinson

For this issue we have two complete solutions to a couple of excellent games: Dallas Quest and Hacker. I have also thrown in a few 'mini hints'. As I have mentioned previously, in future issues we will be seeing some of the best adventures available in the PD. Watch this space!

I must here thank Denon of Israel for submitting the following solutions (solved by EMLYN). Once again one or two of us can sleep easy knowing these two brain teasers have finally been put to rest.

First is Dallas Quest

GET BUGLE,E,GET ENVELOPE,W,W,W,W,PLAY BUGLE,E,E,N,GET GLASS,N,GIVE GLASS,GO BARN,DROP OWL,GET SHOVEL,S,S,S,W, W,DIG,LOOK,READ STONE,E,N, OPEN DESK,DROP MONEY,GET POUCH,N,N, W,W,N,LOOK PLAN,GIVE TICK,GET SACK OPEN SACK,PUT PHOTO IN SACK,GET PARA,JUMP,OPEN POUCH,GIVE TOBACCO, CLOSE POUCH,DROP RING,CLOSE SACK,DROP PARA,GET PHOTO,S,S,S,LOOK PARROT,TICKLE ANACONADA,S,S,GO DINGHY,OPEN POUCH,GIVE TOBACCO,CLOSE POUCH,ROW DINGHY,PLAY BUGLE,S,OPEN POUCH,GIVE TOBACCO,CLOSE POUCH,PULL CURTAIN,DROP BUGLE,DROP SHOVEL,OPEN SACK,DROP POUCH,DROP PHOTO,GET MIRROR,DROP MIRROR,CLOSE SACK,DROP SACK,GET FLASH,CLIMB LADDER,LIGHT FLASH,DROP FLASH,E,S,GET SACK,CLIMB LADDER,GET FLASH,W,UNLIGHT FLASH OPEN SACK,GET PHOTO,SHOW PHOTO,GET COCONUTS,W,GET POUCH,OPEN POUCH, GIVE TOBACCO,GIVE EGGS,GET MIRROR,GIVE MIRROR,GET RING,WAVE RING,HEAT EGGS,LIGHT FLASH,DROP RING,GET MAP,N,GIVE MAP TO SUE ELLEN

As easy as that!

Secondly, we have some HACKER hints which I'm particularly glad about as I still haven't cracked this one!

These are the sequences of which cities to go to first and what to buy and trade when you get there:

1. Switzerland - Give 5,000 - Buy Both Items.

2. Egypt - Give Watch - Buy Both Items.

MARK's GAMES COLUMN continued

3. Rome - Give Tut - Buy Nothing.

4. India - Give Scarab - Buy Nothing.

5. New York - Give Swiss Chalet - Buy Bonds.

6. Tokyo - Give Bonds - Buy Both Items.

7. China - Give Pearls - Buy Jade.

8. Cuba - Give Jade - Buy Nothing.

9. London - Give Camera - Buy Beatles Album.

10. San Francisco - Give Beatles Album - Leave to Washington.

11. Washington - Give F.B.I. Agent Document.

These are the four security checks Hacker will ask for in order.

1.MAGMA, LTD.

2.AX-0310479

3.HYDRAULIC

4.AUSTRALIA

Now to start the game. When it asks for logon type:

LOGON:Australia

It will then take you right in to the game avoiding clicking the mouse on the various items. Next center your time zone so you have daylight on the most populated area. I will now give you a chart of the moves to make to get through the game. NOTE: Be very careful when clicking your moves because the count is VERY important one false move and you will not make it to Washington. This game is timed by moves and not by a time limit. Take your time but be accurate and careful.

When it asks for your name type your name fully because it will print a page from the Washington Post with your name in it, so be sure to have an Epson compatible printer connected.

Well here goes. GOOD LUCK!!!!

6 - East,4 - North,Switzerland - Give 5,000 - Buy Both Items,4 - West,1 - North,6 - East,2 - South,1 - East,Egypt - Give Watch - Buy Both Items,1 - East,1 - North,1 - West,Rome - Give Tut - Buy Nothing,1 - East,2 - North,2 - East,3 - South,1 - East,India - Give Scarab - Buy Nothing,1 - West,3 - North,16 - West,1 - South,New York - Give Swiss Chalet - Buy Nothing,6 - West,1 - North,14 - West,2 - South,5 - East,Tokyo - Give Bonds - Buy Both Items,4 - West,China - Give Pearls - Buy Jade,1 - East,3 - South,2 - East,1 - North,4 - East,1 - South,2 - East,1 - South,2 - East,1 - South,8 - East,3 - North,Cuba

MARK's GAMES COLUMN continued

- Give Jade - Buy Nothing,8 - East,2 - North,2 - West,1 - North,3 - East,London - Give Camera - Buy Beatles Album,3 - West,2 - South,12 - West,1 - North,1 - East,San Francisco - Give Beatles Album - Go to Washington,1 - West,1 - South,10 - East,2 - North,2 - West,1 - South,3 - West.

Washington - You Made It !!!!!!

A COUPLE OF MINI HINTS

Some time ago a received a letter asking for some help with Hitch Hikers Guide to the Galaxy. He was having problems with the intelligence test to get through the door. To solve this you need to reconcile two opposite facts - eg to have stomach ache and not have stomach ache at the same time. Sounds like you need a drink of tea (or not)as the case may be!

While I'm on the subject of Infocom (try to stay awake!) have you ever typed 'taste MGCKJK' in Zork II. If not why not? Go try it now!

Another fiendish puzzle is the Rainbow room in Guild of Thieves. To cross you must step on each colour of the rainbow in reverse order. Does the sign WOBNIAR make sense now?

And finally....

To gain that extra strength required in Scapeghost get a friend to push the sundial with you, then take the Gnomon.

So, until next time, good adventuring and keep those hints and tips coming.

GAMES REVIEWS

MICROX by Mark Fenwick

New from Avalon, comes Microx, a puzzle game, with a slight educational theme. Educational only in that you must assemble chemical compounds for each level.

On loading Microx, you'll soon be presented with a well put together title screen accompanied by a lively piece of music. The title screen goes on to give credits to the authors before prompting you to start the game.

Pressing fire begins play and so the first chemical compound to assemble is shown, along with the time limit to achieve this task in. You do not need a great deal of knowledge in the field of Chemistry so don't get too worried! At the beginning of each level the relative compound, ie. Water, in level one, is shown assembled so you know just how to connect the chemical elements.

The playing area takes up the whole screen, the timer isn't shown, but a bleeping sound will start when you're close to running out of time. The screen is set out a little like an empty crossword puzzle with blocks and channels cut out for your chemical elements to move along.

MICROX continued

The elements of the compound are placed in different parts of the screen, also visible is an outlined curser with which you move them.

Playing the game comprises of both thought and speed, in that you must think where best on the screen to build the compound successfully, not forgetting speed as the timer ticks away. Movement is achieved via the joystick, simply place the curser over the piece of compound you wish to move and press fire to activate the it. The element can be sent in one of four directions, not diagonal though, sliding continuously until coming in contact with either the walls of the maze or another element. This adds a great degree of difficulty as certain exits in walls can not be entered, so a fair amount of thought is needed before moving any pieces. Should you lose sight of how the compound should be constructed pressing any key will show the constructed compound and pauses the timer for a little extra thinking time. If you should beat the timer (which is a close call!) and successfully assemble the compound you are then told 'Good Work!' showing your time left and presented with a multiplechoice question regarding the identification of the completed compound. Should you answer correctly you proceed to the next level, however, get it wrong and you'll begin again at the current level. Don't lose hope though with an A,B,C to the multiple choice you're only looking at a maximum of three attempts per level, eh! You're only given the one life so failure to construct the particular compound in the given time results in your pieces exploding! Though you only have the one life the game has infinate continues so it will stay at the highest level uncompleted.

Overall, Microx is yet another excellent puzzle game, with sharp detailed graphics, colour and clever pieces of music. It also has an addictive feel, as you try to get another level completed. The timer however could have done with a few more seconds on some levels as it's a bit close. Microx is similar in many ways to Atomit, but with a greater degree of difficulty. I'm sure Microx will appeal to both young and old, believe me, even the wife enjoyed this one! Microx is definately not a game you'll complete soon! Microx is currently available from Micro Discount for £6.95

LASERMAZE

From Ke-Soft, renowned for their puzzle type games, comes Lasermaze.

Lasermaze is a puzzle game in which you must destroy a given amount of bombs with a laser, with you playing the part of a Monk (why a Monk, I don't know) you must move blocks around the screen in order to deflect the laser beam.

On booting Lasermaze you'll soon be presented with the title screen, accompanied by a RIAAW piece of music (Rather Irritating After A While) At the bottom of the screen is a list of options, Start begins play, Select allows you to enter a level code, these codes are given on each level so you don't have to start from the beginning each time you reload the game. Option takes you to a construction kit which allows the creation of your own levels!

LASERMAZE continued

The play area takes up most of the screen, while a small section of the right hand side of the screen shows number of shots left. The laser is shown in the middle of the screen represented by a snake type symbol, below this is your character, the Monk. The screen has a border, which is split into blocks of stone which deflect the laser beam while other areas absorb the beam. Scattered around the play area are the bombs and blocks, the bombs stay in a fixed position while the blocks are moveable.

The idea of the game is to destroy all the bombs on each level by firing the laser beam at them. The beam fires in a diagonal direction until either coming in to contact with a bomb which will explode, or hitting a block which will deflect the beam in another direction. When you start off simply press fire to see where the beam goes after deflecting off of blocks, obviously the beam will not bounce around blowing up all the bombs, this would be to easy! So guiding your Monk via the joystick you must push the moveable blocks to where they'll cross the beam Once you're happy with where you've placed your blocks, press fire. The laser will fire a beam and deflect off your blocks and hit a bomb, if you've lined things up right that is, there's a limited amount of shots so plan carefully. Each time you fire a beam the beam will stay visible until you move the joystick, giving you time to plan your next move. With each shot of the laser an icon from the right hand side of the screen will dissapear to show shots left, when you run out of shots your Monk will suffer a death in the form of Spontaneous Human Combustion and you must start the level again. Your Monk is safe touching the Bombs and standing in the way of the laser beam, so it's just a case of carefull planning.

As the levels progress so does the difficulty, the levels become more maze like and more thought is needed to deflect the beam to hit the bombs. In later levels there are other icons which when hit give different results such as more shots symbols and destructive symbols which kill when hit. There are fifty levels to go at and should you eventually complete all fifty you can opt for the construction kit. This can be selected from the main menu and allows you to construct your own screens, placing the different icons around the screen is easily achieved via the joystick.

Overall Lasermaze is very addictive a great puzzler, with the level select easing the frustration of playing from the beginning each time. There's a good degree of difficulty so there's certainly a good few late nights with this one! The addition of a construction set means even longer lasting fun as the only limitation then is your own imagination! Lasermaze is available from Ke-Soft direct priced £8.50 or contact Derek at Micro Discount.

TWAUG
8-BIT

T.W.A.U.G.
P.O.BOX No.8
WALLSEND
TYNE & WEAR
NE28 6DQ

PARALLEL BUS REVEALED

Part III:

Building the serial I/O board.

by EARL RICE

In the first two parts of this series, we learned the basic concepts of the Atari Parallel Bus Interface. This issue, we'll start the actual design of a serial I/O device for the PBI. We'll choose our devices and set up the logic to allow the computer to talk to the 2K ROM or the USART that we'll use for I/O. In the next issue, we'll design address decoders and put software into the ROM to make things work.

Figure 1 is a block diagram of the devices we'll work with this issue. Notice that the signals coming into the select logic are the same ones we invented on our block diagram in the last issue. The exception is RST which comes directly from the PBI connector. The 2K ROM is a 2716, available from almost any surplus house. Be sure to get the 350ns version or it will be too slow for your computer.

The USART and Baud Rate Generator are from Radio Shack. See the parts list for catalogue numbers. I picked this USART because it is readily available. It is also simple to design with because it only has four registers to deal with, and all are brought out to IC pins. That means we can hand-wire some functions and save writing unnecessary software. In the next issue we'll explain how you can make the circuit more programmable if you prefer to.

For this example, however, we'll hand-wire the control register to give us 300 baud, 7 data bits, no stop bits and no parity. Figure 3 is a inscription of USART pin functions and has the information you need to change the data format. Figure 4 and it's associated table show how to change baud rate. Note that the baud rate generator has to run at 16 times the baud rate you want from the USART.

The select logic bears some discussion. Because PBI timing requirements are tight, we need to use fast logic chips to be sure things work. To make matters worse, the PBI can electrically drive only one low power TTL load. When we decode addresses, we'll either need to tie two gate imputs to some lines and overload them, or put a low power buffer on the line and add an extra gate delay to our circuit. Neither alternative is very attractive.

Fortunately, there is a logic family available that combines the loading characteristics of CMOS with the speed of Schottky TTL. This combination of high speed with virtually no DC load on the PBI lines is just what we are looking for. The logic family is the 74HCTXX series. These are the parts to use here. They are exactly function and pin compatible with TTL.

Be sure to get 74HCT parts and not 74HC parts. The HTC series is a little scarce on the hobby market, but they are available.

Figure 2 is a schematic diagram of our serial I/O device.

PARALLEL BUS continued

Notice that the address lines to the 2716 ROM are left off. This is to avoid clutter. We'll put them in the next issue when we do address decoding. ICs 4 and 5, the NAND and NOR gates, are the select logic.

The 2K ROM is selected when the signal from the DEVICE ENABLELATCH is LOW AND D8XX-DFXX is LOW. Follow the path through the two NOR gates. You'll notice that the second one is used as an inverter. LOWs at both pins 4 and 6 produce a LOW at pin 13, giving CHIP SELECT (CS) to the ROM.

This doesn't allow the ROM to be read, however, because the POWER DOWN (PD) line has to be brought LOW to enable the ROM outputs. The R/W signal does that every READ cycle. When it brings pin 9 off IC-4 HI, pin 10 goes LOW, enabling the ROM outputs. When both CS and PD are LOW, the ROM is on the bus.

We use R/W for the PD signal because its state is set at the beginning of the 6502 machine cycle, and the PD input takes about 250ns to work. If we waited for address decoding, a slow ROM might not come on quickly enough. CS operates in less than 30ns, so there's plenty of time available to wait for decoding and device enable to happen.

The USART is set up to operate as a single read or write register. Any address from $D100 to $D1FF will enable the USART. This wouldn't do at all if we wanted to program its control functions or read its status register. But we've hand-wired those functions for our example, so it really doesn't matter. Besides, it saves parts cost.

In the next issue we'll deal with embellishments. For now, writing to any address in the $D1XX range puts a character in to the transmit register and the USART will send it. Reading any address in that range reads the last character received by the USART. The DS1-DS8 pins go to the transmit register, and the RD1-RD8 pins go to the receive register. We've wired these together and connected them to the data bus so the computer can write and read USART data.

When the signal from DEVICE ENABLE LATCH is LOW at pin 3 of IC-4 AND the $D1XX signal is LOW at pin 3 of IC-4, its output goes HIGH and enables the read-write gates from IC-5. Then if R/W is HIGH at pin 1 of IC-5, pin 3 goes LOW, selecting READ DATA enable (RDE) and placing the USART receive register on the bus to be read.

At the same time, pin 10 of IC-4 brings pin 12 of IC-5 LOW keeping pin 11 HIGH so the DATA STROBE (DS) of the USART is disabled. (Why isn't the ROM selected too? because pin 13 of IC-4 is HIGH.) If R/W were LOW, pin 1 of IC-5 would be LOW and RDE would be disabled while pin 12 of IC-5 would be HIGH and DS would be enabled.

So that's how the select logic works. The only new signal we have is RST which comes from the PBI bus to reset the USART whenever the computer is reset. We send the buffered signal back out as DEVICE RESET (DRST) to reset the device enable latch. we'll see how that works in the final article.

In the meantime, you might want to go about scrounging parts. The circuits can be built using wire-wrap boards if you want. I prototyped on a perforated bread board and it worked fine. Leave room for another half dozen 14 pin gate IC's, a 50 pin ribbon cable header, a 9 pin D-type connector (for I/O), a 5V power connector, and a little extra for any enhancements you might want.

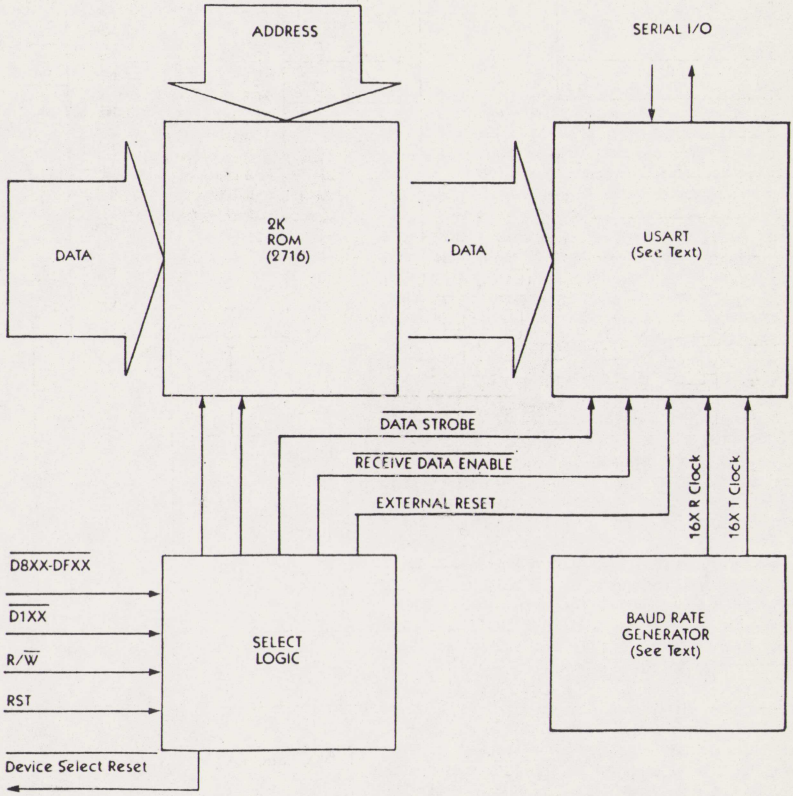
In the next issue, we'll wrap things up with the address decoding logic, software drivers, and some suggestions for your own enhancements. See you then!

PARTS LIST

IC-1 baud rate Generator	Radio Shack Cat. No. 276-1795
IC-2 USART	Radio Shack Cat. No. 276-1794
IC-3 EPROM	2716-1 (350ns or faster)
IC-4 Quad 2-input NOR	74HCT02
IC-5 Quad 2-input NAND	74HCT00
CRYSTAL 5.066MHz	

PARALLEL BUS continued

Figure 1. I/O Device Block Diagram



PARALLEL BUS continued

Figure 2. I/O Device Schematic

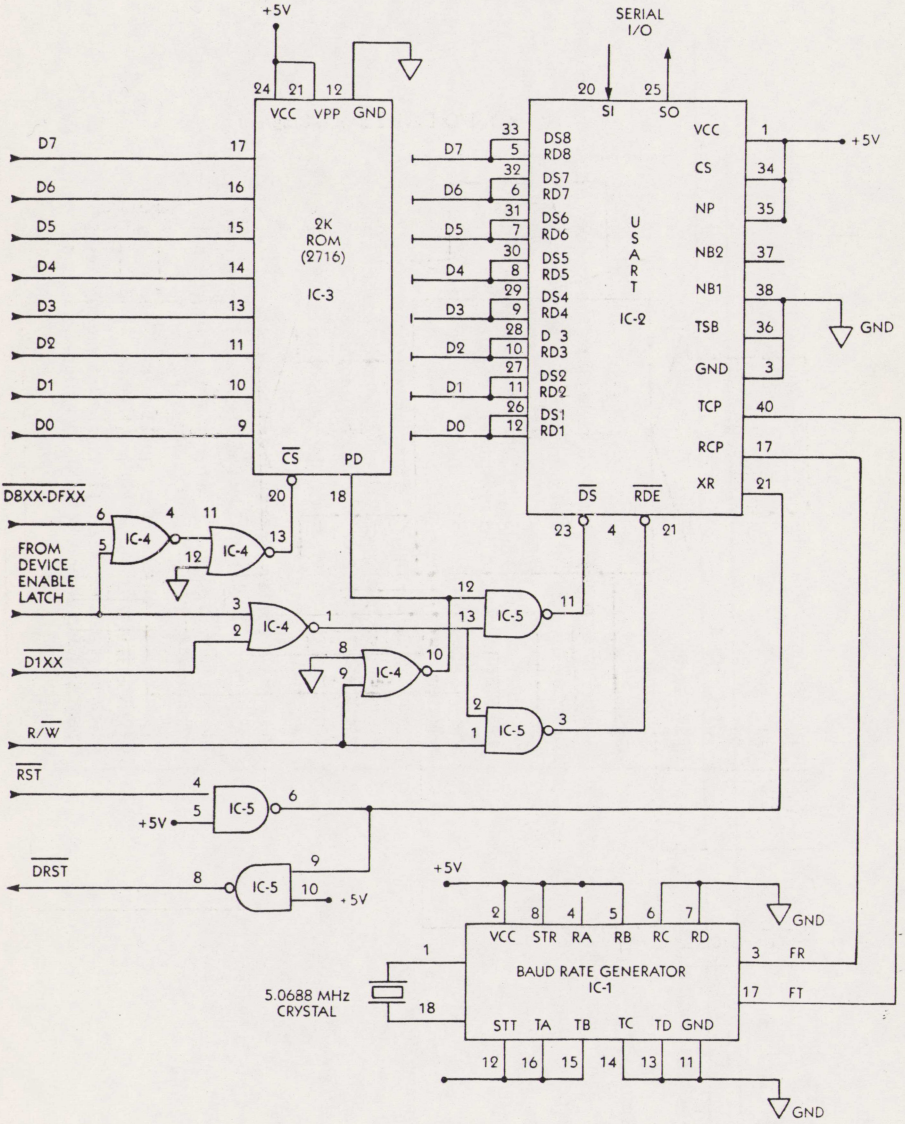


Figure 3. UART Pin Functions

PIN FUNCTIONS

PIN	NAME (SYMBOL)	FUNCTION
1	V_{CC} Power Supply (V_{CC})	+5V Supply
2		Not connected
3	Ground (V_{GI})	Ground
4	Received Date Enable (RDE)	A logic "0" on the receiver enable line places the received onto the output lines.
5-12	Received Data Bits (RD8-RD1)	These are the 8 data output lines. Received characters are right justified: the LSB always appears on RD1. These lines have tristate outputs, i.e., they have the normal TTL output characteristics when RDE is "0" and a high impedance state when RDE is "1". Thus, the data output lines can be bus structure oriented.
13	Parity Error (PE)	This line goes to a logic "1" if the received character parity does not agree with the selected parity. Tri-state.
14	Framing Error (FE)	This line goes to a logic "1" if the received character has no valid stop bit. Tri-state.
15	Over-Run (OR)	This lines goes to a logic "1" if the previously received character is not read (DAV line not reset) before the present character is transferred to the receiver holding register. Tri-state.
16	Status Word Enable (SWE)	A logic "0" on this line places the status word bits (PE, FE, OR, DAV, TBMT) onto the output lines. Tri-state.
17	Receiver Clock (RCP)	This line will contain a clock whose frequency is 16 times (16X) the des[illegible] receiver baud.
18	Reset Data Available (RDAV)	A logic "0" will reset the DAV line. The DAV F/F is only thing that is reset.
19	Data Available (DAV)	This line goes to a logic "1" when an entire character has been received and transferred to the receiver holding register. Tristate-Fig. 16
20	Serial Input (SI)	This line accepts the serial bit input stream. A Marking (logic "1") to spacing (logic "0") transition is required for initiation of data reception. Fig. 15, 16.
21	External Reset (XR)	Resets all registeres except the control bits register. Sets SO, EOC and TBMT to a logic "1". Resets DAV and error flags to "0". Clears input data buffer. Must be tied to logic "0" when not in use.

22	Transmitter Buffer Empty (TBMT)	The transmitter buffer empty flag goes to a logic "1" when the data bits holding register may be loaded with another character. Tri-state. See Fig. 9, 11.
23	Data Strobe (DS)	A strobe on this line will enter the data bits into the data bits holding register. Initial data transmission is initiated by the rising edge of DS. Data must be stable during entire strobe.
24	End of Character (EOC)	This line goes to a logic "1" each time a full character is transmitted. It remains at this level until the start of transmission of the next character. See Fig. 8, 10.
25	Serial Output (SO)	This line will serially, by bit, provide the entire transmitted character. It will remain at a logic "1" when no data is being transmitted.
26-33	Data Bit Inputs (DB1-DB8)	There are up to 8 data bit input lines available.
34	Control Strobe (CS)	A logic "1" on this lead will enter the control bits (EPS, NB1, NB2, TSB, NP) into the control bits holding register. This line can be strobed or hard wired to a logic "1" level. See Fig. 19.
35	No Parity (NP)	A logic "1" on this lead will eliminate the parity bit from the transmitted and received character (no PE indication). The stop bit(s) will immediately follow the last data bit. If not used, this lead must be tied to a logic "0".
36	Number of Stop Bits (TSB)	This lead will select the number of stop bits, 1 or 2, to be appended immediately after the parity bit. A logic "0" will insert 1 stop bit and a logic "1" will insert 2 stop bits. The combined selection of 2 stop bits and 5 bits/character will produce 1½ stop bits.
37-38	Number of Bits/Character (NB2, NB1)	These two leads will be internally decoded to select either 5, 6, 7 or 8 data bits/character. NB2 NB1 Bits/Character 0 0 5 0 1 6 1 0 7 1 1 8
39	Odd/Even Parity Select (EPS)	The logic level on this pin selects the type of parity which will be appended immediately after the data bits. It also determines the parity that will be checked by the receiver. A logic "0" will insert odd parity and a logic "1" will insert even parity.
40	Transmitter Clock (TCP)	This line will contain a clock whose frequency is 16 times (16X) the desired transmitter baud.

PIN FUNCTIONS

PIN NO.	SIGNAL	FUNCTION
1	XTAL/EXT1	Input is either one pin of the crystal package or one polarity of the external input.
2	V_{CC}	Positive power supply—normally +5V
3	f_R	This output runs at a frequenc selected by the Receiver divisor select data bits.
4-7	R_A, R_B, R_C, R_D	These inputs, as shown in Table 1, select the receiver output frequency, f_R.
8	STR	A high level input strobe loads the receiver data (R_A, R_B, R_C, R_D) into the receiver divisor select register. This input may be strobed or hard-wired to a high level.
9	NC	
10	NC	
11	GND	Ground
12	STT	A high level input strobe loads the transmitter data (T_A, T_B, T_C, T_D) into the transmitter divisor select register. This input may be strobed or hard-wired to a high level.
13-16	T_D, T_C, T_B, T_A	These inputs, as shown in Table 1, select the transmitter output frequency f_T
17	f_T	This output runs at a frequency selected by the Transmitter divisor select data bits.
18	XTAL/EXT2	This input is either the other pin of the crystal package or the other polarity of the external input.

Figure 4.

REFERENCE FREQUENCY = 4.915200MHz

Divisor Select DCBA	Desired Baud Rate	Clock Factor	Desired Frequency (KHz)	Divisor	Actual Baud Rate	Actual Frequency (KHz)	Deviation
0000	50.00	16X	0.80000	6144	50.00	0.800000	0.0000%
0001	75.00	16X	1.20000	4096	75.00	1.200000	0.0000%
0010	110.00	16X	1.76000	2793	109.93	1.758983	0.0100%
0011	134.50	16X	2.15200	2284	134.50	2.152000	0.0000%
0100	150.00	16X	2.40000	2048	150.00	2.400000	0.0000%
0101	300.00	16X	4.80000	1024	300.00	4.800000	0.0000%
0110	600.00	16X	9.60000	512	600.00	9.600000	0.0000%
0111	1200.00	16X	19.20000	256	1200.00	19.200000	0.0000%
1000	1800.00	16X	28.80000	171	1796.49	28.743859	0.1949%
1001	2000.00	16X	32.00000	154	1994.81	31.916883	0.2697%
1010	2400.00	16X	38.40000	128	2400.00	32.000000	0.0000%
1011	3600.00	16X	57.60000	85	3614.11	57.825882	0.3921%
1100	4800.00	16X	76.80000	64	4800.00	76.800000	0.0000%
1101	7200.00	16X	115.20000	43	7144.19	114.306976	0.7751%
1110	9600.00	16X	153.60000	32	9600.00	153.600000	0.0000%
1111	19200.00	16X	307.20000	16	19200.00	307.200000	0.0000%

REFERENCE FREQUENCY=5.068800MHz

Divisor Select DCBA	Desired Baud Rate	Clock Factor	Desired Frequency (KHz)	Divisor	Actual Baud Rate	Actual Frequency (KHz)	Deviation
0000	50.00	16X	0.80000	6336	50.00	0.800000	0.0000%
0001	75.00	16X	1.20000	4224	75.00	1.200000	0.0000%
0010	110.00	16X	1.76000	2880	110.00	1.760000	0.0000%
0011	134.50	16X	2.15200	2355	134.52	2.152357	0.0166%
0100	150.00	16X	2.40000	2112	150.00	2.400000	0.0000%
0101	300.00	16X	4.80000	1056	300.00	4.800000	0.0000%
0110	600.00	16X	9.60000	528	600.00	9.600000	0.0000%
0111	1200.00	16X	19.20000	264	1200.00	19.200000	0.0000%
1000	1800.00	16X	28.80000	176	1800.00	28.800000	0.0000%
1001	2000.00	16X	32.00000	158	2005.06	32.081013	0.2532%
1010	2400.00	16X	38.40000	132	2400.00	38.400000	0.0000%
1011	3600.00	16X	57.60000	88	3600.00	57.600000	0.0000%
1100	4800.00	16X	76.80000	66	4800.00	76.800000	0.0000%
1101	7200.00	16X	115.20000	44	7200.00	115.200000	0.0000%
1110	9600.00	16X	153.60000	33	9600.00	153.600000	0.0000%
1111	19200.00	16X	307.20000	16	19800.00	316.800000	3.1250%

TEXTPRO WORD PROCESSOR

by Max Gerum

I thought that I wouldn't have to write anything else about TextPro in this issue but I was wrong. There are still a number of commands I haven't covered.

Before I start writing my bit this week, I would like to share a little prayer with you if I may. I read it somewhere but cannot remember where, I racked my brain, (I believe I've got something like that somewhere), to remember where I'd seen it but to no avail. Whilst looking through some papers I came across this prayer again. I find it very interesting and I have no doubt that you will too. The first time I read it I didn't think that I would share it with anyone else, as I wasn't writing for a newsletter at the time. As mentioned above I am unable to recollect the authors name or the magazine or whatever it was published in, so please accept my apology.

Have you ever wondered what would happen, if we ever get a power cut or a heavy power surge, to your computer memory after you have sat typing in an article, of maybe three or four pages long and you haven't saved it to disk? I dare not think about it, because my computers memory is always full of articles ready to be printed out.

Here is the prayer I want to share with you and it goes something like this:

Our Program who art in Memory, "Hello" by Thy Name, Thy Operating System come, Thy Commands be done, at the Printer as it is on Screen, Give us this our daily Data and forgive us our I/O Errors as we forgive those whose Logic Circuits are faulty. Lead us not into frustration and deliver us from Power Surges. For Thine is the Algorithm, the Application and the Solution, looping forever and ever. RETURN.

Now that I've said my prayer and also shared it with you I can write my piece in peace.

We are still with TextPro. When you compare TP with any other word processor you will find that when you're in the editor you have 23 lines on the screen at once to which you can write to and you have only one command line. There are some word processors where half of the screen is taken up with icons and other command lines. For instance when you want to save your text or print it, there is no need to go to another menu to perform these functions in TextPro, not like some other WProcessors. All you need to do is press the appropriate command keys for save, load or print, whilst you are in the editor. You don't have to move from one option to another or icon to icon to perform these tasks. That's why I prefer TextPro, all the functions are quick and easy to perform. If you want to check the directory of your disk, you just press CTRL-M from the editor and the directory is displayed on the screen. From there you can load any text file you wish by highlighting the filename and pressing CTRL-L, the screen changes automatically back to the editor when the text is loaded.

In this issue I want to explain some of the print format commands, in the past issues I only fleetingly commented on these functions. These print format commands are very important to the word processor if you want to set up your text to be printed in a professional manner.

The following print commands allow the user to specify the values of print format parameters. They are defined by lower case inverse keys. I explained in an earlier issue that the inverse keys are entered by holding down SELECT when typing the key.

These commands must be entered prior to the text, they may be entered on a separate line or on the same line as the text, separated by a space between the last command character and the text. But the commands are entered without a space between the inverse command character and the parameter. I will enter some examples to demonstrate the correct usage, the ⟨⟩ characters surrounding a character, means that the character is in inverse.

EXAMPLES:

⟨l⟩10⟨r⟩70 Correct. This will format the text with a left margin of 10 and a right margin of 70.

⟨l⟩10⟨r⟩70Incorrect. No space between the '70' and the start of the text.

⟨l⟩10⟨r⟩70RETURN
Correct. The Return will not be counted as a line for paging.

⟨l⟩10 ⟨r⟩70 This is also correct. The space between the 10 and the right margin command character is perfectly legal.

These are the margin commands, I will now list in alphabetical order all other commands some with default settings and others without defaults. But please remember that all these commands are type in inverse while the parameter numbers are typed in clear print.

⟨b⟩# = bottom margin the default setting is 58. The bottom margin can be changed by just typing a different number after the inverse ⟨b⟩.

⟨d⟩# = down lines, no default setting.

⟨d⟩4 moves down the page 4 lines.

⟨f⟩text = footer, no default setting.

Setting a footer at the beginning of a text will print the footer at the end of the page. I will show you an example after the header command.

⟨g⟩ = goto link file.

Use this command at the end of the text file to specify the next file to print. Used to segment a large document together for printing. For example:

⟨g⟩D1:TEXTPRO.001(RETURN)

TEXTPRO WORD PROCESSOR continued

<h>text = header has no default.

The TextPro manual has been set up to print a header and a footer, when you purchase the TextPro disk from Twaug you also receive the disk with the manual for printing. Take section 1 it has been setup as this example:

<h><c>Section 1 - TextPRO+ Introduction -
<@>1<f><c>Page #

This will print the header centered as "Section 1 - etc. and the footer centered with Page 1. It will print the header on all following pages and will also number the footer following with the follow-on numbers.

<i>text = information.

Using the '<i>' command you can create non-printing notes for your information, when entering text after a Return.

<j> - default off.

Put a <j> in the print file to set linefeeds on for the printed file. This can also be done with the print option selected with the CTRL-; command. The linefeeds can only be turned on with this command, they are not turned off with another <j>.

<k># - lines down without count.

Sends the # lines to the printer without counting them for paging. For instance:

<k>5 sends 5 uncounted CR's to the printer.

<l># - left margin - default 5

This sets the left margin for the document.

<m># - margin outdent - no default.

When used with a # parameter the line is outdented # spaces. For instance:

<m>3<l>8 1. Sets the left margin to 8 and outdents the '1. ' because the parameter for the 'm' command is three. An <m> without a # parameter will send this line all the way out to the left edge of the page.

<n># - next page - no default.

This will force a page break. It ends the current page and printing will start with on a new page. When a # parameter is specified the break is conditional if there are less than or equal to number of lines left on the page.

<o>dec.# - overstrike - default 95 "_"

This specifies the decimal value of the overstrike character sent after the backspace when using the <u>nderline command.

<o>47 <u>this will print the "/" character over all of the characters between the <u> commands.

Please NOTE: Your printer must be able to backspace by sending the backspace character, dec.#8, to the printer.

<p># - page length - default 66

<q>0 = justify off, 1 = justify on - default off

<q>1 Including this command in the first line before your text will right justify all your text.

<q>0 This will give your text a ragged right margin.

<r># - right margin - default 75

This sets the right margin for the text.

<s># - line spacing - default 1. This sets the # spacing between the lines.

<t># - top margin - default 5. This sets the number of blank lines printed at the top of the page. The header is printed on the first line, so a setting of 5 will put 4 lines between the header and the first line of the text.

<w> - page wait - default off. I do not waste paper, therefore this command is typed before I begin typing my text and the prin: will pause at the end of each page and the prompt,

"Insert sheet, PRESS ANY KEY" will appear. This feature allows you to insert a new sheet and/or turn over for printing on both sides.

<x># - column across page - default 80

This specifies the number of characters printed across the page. For pica print with 8 1/2" paper this value is 80. Adjust this parameter for different print pitches.

<y># - zmargin setting - default 5

<@> - starting page number - default 1

Sets the page number to start numbering the printed pages.

<?># - start printing with # - default 1. Sets the # page to begin printing. Include this command at the beginning of your text to tell the printer to begin printing from page #. If you miss this feature out the printer will not start printing.

<?>4 will begin printing the document from page 4 and continue to the end of the file.

<&># - stops printing with # - no default. Sets the # page to stop printing. Use it with <?> command to specify the pages to print. For example:

TEXTPRO WORD PROCESSOR continued

<?>2<&>3 will start printing at page 2 and stops after printing page 3.

<!># - skip between pages - default 0. Used to print every # page. Set to 1 will print every other page. Use this feature to print both sides of the page.

<?>1<!>1<&>20 this example will start printing from page 1 print every odd page between 1 and 20. If you set <?> to 2 will start printing from page 2 and print all the even page.

You can also use this feature to print both sides of the pages for binding. You can adjust the headers and footers with the <e>dge and <z>margin before each printing to set them right and left on facing pages.

<c> - center line. This command will center a header.

<e> - edge line right. This sets a line even with the right margin value.

<z> - insert zmargin - default 5. Places <y># of spaces at the location. This command can be imbedded in the text and can be used multiple times. For instance:

<y>10<z><z><z>This line will be indented 30 spaces.

<u> - underline. Starts and stops, a simple and generic form of underlining, by sending a backspace followed by the underline character for every printed character.

This is <u>underlined<u>.

These are the most used commands in TextPro in your every day use of the word processor. But to get more familiar with TP, the manual should be read.

There is one more feature you can perform with TextPro and that is create and print fancy labels directly from the editor. Have a look at the diagram at the end of Cracking the Code article, that has been set up and printed from the editor.

Ted Skrecky wrote an article about this feature in the X103 newsletter, it is produce by the Garden City Atari Computer Enthusiasts of CANADA.

Most dot-matrix printers have one or more alternative character sets built-in and these can be access by sending special commands to the printer. I wonder if there is anyone who has used a word processor on the 8-bit, typed in some words, drawn some lines around it and sent the result to the printer. Well TextPro lets you do that, TP lets you program your printer to access the other character sets to print out.

Once you've create a label you want to see what it looks on paper. To print the inverse characters which you use for creating a label, you must write the buffer content to the printer. Use the CTRL-S function for saving to disk, now backspace out the 'D:' in the command line and type 'P:' and press Return, make sure your pinter is on line.

To switch your printer in to the Graphics mode consult your manual. I am only able to tell you how to switch a Citizen 120D and 120D+ also a Star LC24-100 into that mode. To switch the 120D to the IBM Graphics mode set switches 3 and 4 to ON, that's all that needs to be done to print labels direct from the TP editor. The Citizen 120D+ is just as easy, the only difference is to set switches 2 and 3 to ON. To switch the Star LC24-100 to that mode you must first enter the (EDS) mode and by using the FONT button go to the D Bank and set switch 1 to ON, make sure you save this setting.

When using a Citizen printer you don't need to choose a character set, but you could type onto the first line the ESC and upper case E for emphasized printing, this way your label is more noticeable, and press Return. By switching D1 to on, I've set my Star LC24-100 into the Graphics mode. I must also tell the printer which character set I want to use. The Star has two sets built-in I chose set #2, to activate this set I type on the first line of the TP editor an "ESC and 6". And to make the label more noticeable I also enter an ESC and E for emphasized printing. I would also like to suggest to you that you tell your printer to print in one direction only (uni-directional) for increased accuracy, enter ESC U1, or set switch A5 to OFF.

The characters you're looking for should be in your printer manual. We are using the inverse characters, Decimal Codes 176 to 218. To create a label use the printed "raw" example below as a guide but enter the characters forming the lines of the label in inverse. I wasn't abel to print the example in inverse as I wanted to, but at least it shows you how to create a label.

About 34 characters wide and 7 characters high will fit on a label of 4" wide and 1 1/2" high with about a 1/4 of an inch space all round.

```
IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM;
:          TWAUG NEWSLETTER       :
GDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD6
: SIDE A: PUBLIC DOMAIN LIBRARY   :
: SIDE B: UTILITY PROGRAMMES      :
GDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD6
:     PLACE YOUR ORDER TODAY      :
HMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM<
```

TWAUG NEWSLETTER
SIDE A: PUBLIC DOMAIN LIBRARY SIDE B: UTILITY PROGRAMMES
PLACE YOUR ORDER TODAY

TEXTPRO WORD PROCESSOR continued

Here is another example a label can be set up. This is actually a much easier way to create the label, just type all zero's in inverse and the result is shown below.

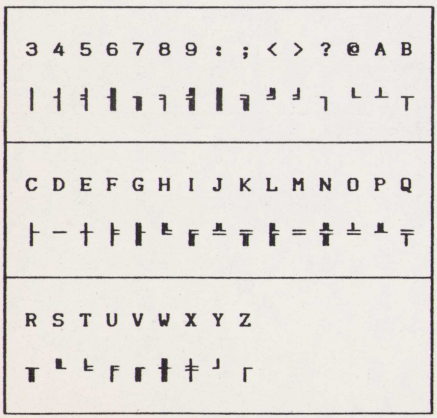
```
0000000000000000000000000000000000
0  TYNE & WEAR ATARI USER GROUP  0
0000000000000000000000000000000000
00000 PUBLIC DOMAIN LIBRARY 000000
0000000000000000000000000000000000
0 SIDE A: 000 UTILITY PROGRAMMES 0
0 SIDE B: 000 PRINT SHOP ICONS   0
0000000000000000000000000000000000
```

```
TYNE & WEAR ATARI USER GROUP
PUBLIC DOMAIN LIBRARY
SIDE A:   UTILITY PROGRAMMES
SIDE B:   PRINT SHOP ICONS
```

What about something like this below this text, there are lots of different ways a label can be created. There is one consolation, being able to use a word processor gives you a quicker way to print labels, especially when you're in a hurry.

TYNE & WEAR ATARI USER GROUP	
PUBLIC DOMAIN LIBRARY	
SIDE A:	UTILITY PROGRAMMES
SIDE B:	PRINT SHOP ICONS

Here is a printout of all the characters used to create labels. Typing the characters in inverse will print the single and double lines.



DISK CONTENT

COPYPLUS: is a multi-purpose copy utility that will copy most files including cassette files. The instructions are on the disk, these give you the full operating procedure and also tell you how to read the length of a file in bytes. A very handy utility for your library.

CONVERT.BAS: is a BASIC little program to convert all Imperial weights and measures to Metric and vice versa and lets you also convert the U.S. measures to metric and back. The program is menu driven and self-explanatory, instructions on disk.

PSPRINT.OBJ: this is a Print Shop printing utility, it is easy to follow, no seperate instructions on disk. To load and run use 'L' Option from DOS,Basic must be disabled.

BSPDTIME.BAS: this BASIC program will convert your SpartaDOS 3.2 date-time to British standard. Read the article starting on page 32, it tells you everything you need to know.

ROMAN.BAS: is a little program to convert any number into Roman numerals.

ALIENS: load this pin ball game with the 'L' Option from DOS without BASIC.

TINYMITE.COM: is a shoot them up game, load with the 'L' Option from DOS with BASIC disabled. You need to be quick to survive because you are being bombarded all the time.

SIDE B of the disk is an interesting utility as a BONUS. TURBODUP has its own documentation which can be read from the menu with the 'F' Option. This utility performs many options just like a DUP.SYS but it is different. For instance, using the 'H' option you can map a disk, or by using 'J' option and entering an address you can find a label in decimal or in hex. An interesting function is when using the 'K' option, this lets you find a 6502 operation code. Entries may be made in decimal, hex, binary, Atascii or mnemonics. A very usefull program. Enjoy this issue disk.

A PUBLIC DOMAIN REVIEW

by Bill Jackson.

This is a review of a program that T.W.A.U.G. is adding to their Public Domain library.

The program is by 'The Software Factory' (Birmingham) and was written by Russell May. The program is entitled 'CHARACTERSET ANIMATOR EDITOR'.

Right, lets get started, as you may have gathered this is a character set editor program, that allows you to load, edit and animate several character sets.

Yes!, several character sets. You can load up to 8 (eight) sets on a standard Atari 800XL and freely work between them, coping individual or multiple characters (up to 32 at a time).

The program was written in 'ACTION' so is very quick and keyboard responsive. There is also a neat little 'TUTORIAL' program on the disk that you can load from the title screen. Note:- The tutorial is for the screen only, i.e. there is no facility to print-out a hard-copy. This is no real problem though, as the program is very user friendly. If you DO get stuck you could always re-boot the disk and go through the tutorial taking notes where need be.

Lets now take a look at the disk contents to see what else is on the disk:-

DOS	SYS	037	DUP	SYS	042
RAMDISK	COM	009	AUTORUN	SYS	027
EDITOR	EXE	117	TUTORIAL	EXE	091
TUTOR	CHS	009	TITLE	DMP	024
EDIT25	ACT	093	CHEDIT	BAS	016
YAHTZEE	BAS	053	YAHTZEE	FNT	009
YAHTZEE	DAT	009	WIDETEXT		009
MODERN		009	FUTURE		009
ROUND		009	FONT4		009
GTIA		009	H1		009
H2		009	H3		009
H4		009	H5		009
T1		009	T2		009
T3		009	T4		009
T5		009	T6		009
T8		009	T7		009

RAMDISK.COM For use with 130XE/upgraded 800XL's.
AUTORUN.SYS Was originally named TITLE.EXE
EDITOR.EXE The main prog.
TUTORIAL.EXE The tutorial prog.
TUTOR.CHS A re-defined character set.
TITLE.DMP Used in intro.
EDIT25.ACT The ACTION source file.
CHEDIT.BAS A demonstration BASIC prog to let you see what can be done with your re-defined sets.

YAHTZEE.BAS A basic game that uses 'sets'.

A PUBLIC DOMAIN REVIEW continued

The rest (apart from Yahtzee related files) are standard 9 sector fonts (I normally give these a .FNT extender).

H1-H5 Are five fonts of horses, very impressive when animated.
T1-T8 Are eight 'test' sets, equally impressive when animated.

Now that you know what's on the disk, let's get back to the 'EDITOR' program.

Boot up the disk (without basic) and at the title screen, press OPTION. This will load the 'TUTORIAL' program. It doesn't take long to get through the Tutorial and then you're ready to re-boot and take a look at the EDITOR.

On re-booting and pressing SELECT to load the Editor, you will see the 'Screen Layout', as follows:-

```
*************************************************
*                                               *
*    CASE:H1/LO  WRAP:OFF/ON   ASC:  INT:  KEY: *
*                                               *
*        **********                             *
*        *        *          MENUS              *
*        *  EDIT  *           GO                *
*        *        *          HERE               *
*        *  GRID  *                             *
*        *        *                             *
*        **********                             *
*  ----------------------------------------     *
*          THIS IS THE COMMAND LINE             *
*  ----------------------------------------     *
*                                               *
*    *******                                    *
*    * SET *  NORMAL ATARI CHARACTER SET        *
*    *  1  *                                    *
*    *******  THIS IS FOR REFERENCE ONLY        *
*                                               *
*  ----------------------------------------     *
*        THIS IS THE TEST LINE                  *
*  ----------------------------------------     *
*                                               *
*         THIS IS WHERE YOUR WORK SET GOES      *
*         ALL CHANGES TAKE PLACE ON THIS SET    *
*         AND NOT ON THE ABOVE ATARI SET        *
*                                               *
*************************************************
```

This is the MAIN MENU, which you will find at the top right of the screen

```
[E] EDITOR            [A] ANIMATE
[L] LOAD SET          [T] TEST LINE
[S] SAVE SET          [R] RESTORE
[C] COPY SET          [G] GTIA MODE

[X] EXCHANGE(sets)    [K] COLOURS *
```

A PUBLIC DOMAIN REVIEW continued

```
[M] MOVE-CHS          [B] BLANK SET
[D] DISK              [H] HEX/DEC *
```

Pressing [ESC] will normally get you out of trouble if you select the wrong item. At this point the best thing to do is choose option [L] and LOAD a font file from disk. If you don't load a font file and go straight to the EDITOR MENU, you will not be able to 'edit' characters. You would in fact be defining characters from scratch (not easy) it's much easier to alter existing font files.

Load any 9 sector font file and start experimenting. Most options are self explanatory. Choosing option [T] TEST LINE allows you to type any character to the test line to see what it looks like in graphics mode 1 & 2. Pressing OPTION toggles between LOWERcase and UPPERcase.

Choosing option [R] RESTORE, replaces the character set of your choice with the standard ATARI character set.

Option [M] MOVE-CHS (my favourite) allows you to copy (move) 1 to 32 characters from one set to another. Just follow the on-screen prompts.

Choosing option [D] DISK, allows access to the DISK SUB-MENU. Which looks like this:-

```
:     DISK UTILITY    :
: CURRENT DRIVE IS 1 :

  D for DIRECTORY
1-8 to change drive
ESC return to main menu
  S format SINGLE
  F format DOUBLE
```

Note:- Option F formats in what is sometimes called ENHANCED density (Atari DOS 2.5).

If you wish to change the drive number for future LOADS & SAVES then this is the place to do it.

Pressing ESC takes you back to the MAIN MENU.

Choosing option [E] EDITOR, takes us to yet another menu, namely the EDITOR MENU.

Which looks something like this (I've omitted the arrow symbols in order to save my printer going haywire):-

```
                EDITOR  MENU

E: EDIT         C: CLEAR          W: WRAP
LEFT(ARROW)     RIGHT(ARROW)      UP(ARROW)
DOWN(ARROW)     X: ROTATE         T: TRANSFER TO
S: SWAP         U: UPSIDEdown     F: TRANSFER FROM

          R: RESTORE        M: MIRROR
```

A PUBLIC DOMAIN REVIEW continued

ESC: MAIN MENU I: INVERT

Options K(Colours) & H(Hex/Dec) from the MAIN MENU can be used from this menu also.

Again, most of the options in the EDITOR MENU are self explanatory so I'll just mention a couple.

Option E: EDIT toggles between selecting a character to edit (use joystick to move around set and press trigger to select character) and plotting in the character grid.

With WRAP OFF (default), using any of the arrow keys will move the character OFF the editing grid and this data will be lost. Perhaps it is adviseable to set WRAP to ON, at least until you get used to using the editor.

Pressing R: RESTORE replaces the current character that you are working on with the standard Atari equivelant. It does NOT restore the original character that you were editing.

Once you have mastered the basics of the program you can LOAD in the five HORSE files (H1-H5) and use the [A] ANIMATE option in the MAIN MENU to see what can be achieved with this program. You will also want to load-in the eight test files (T1-T8). These are the eight fonts used in CHEDIT.BAS to create an impressive title screen.

In conclusion, I found this to be the best program of its kind that I have seen. I used to use ANTIC's ENVISION character set editor but in future I'll be using 'THE SOFTWARE FACTORY'S' CHARACTERSET ANIMATOR EDITOR.

The ability to work on up to 8 files at once, plus the ease of use, makes this an exceptional piece of software which would suit beginners and advanced Atari users alike. An absolute snip at a Public Domain price, I highly recommend this program.

GOOD NEWS!!!

We now have the prototype finished copy of the new book based on MAPPING THE ATARI. We are at present looking at different ways for presenting the book and at the same time keeping the cost as low as we possibly can.

We are hoping to have the book ready by mid-April for the SPRING ALL MICRO SHOW at Bingley Hall. If we cannot manage to get enough books ready for sale we are hoping to have at least one or two books with us for visitors to our stand to have a browse through it. We do not believe that any purchaser will be disappointed with the book.

Everyone who ordered in advance by post will be notified by letter as soon as we have the book ready for sale.

DAISY DOT III AND PAGE 6 WRITER

A SMALL CORRECTION

After I submitted the article to be printed in the last newsletter, I made some improvements to it and sent a modified version about a week later. I also sent a copy to Bill Jackson in Scotland who spotted a mistake and pointed it out to me. (Thanks Bill). I then telephoned the correction to David Ewens. He was out so I had to leave a message. I am mentioning all this so that you can see that I was lucky to get the thing printed at all. (Is anyone else as much trouble I wonder).

It is therefore quite understandable that a small error was made in the printing of the START.BAT file; the one printed was from the original write-up. The last entry should be on two lines as follows:

D8:
WP

As it was shown, the system will work, but altering it as above makes D8: the default and makes the whole system much easier to use.

Another very small point (nitpicking?) is that I wrote the title to say 'Max emulation' not 'Max's emulation'; the idea being a subtle play on the word 'Max', as both Mr. Gerum's christian name and an abreviation for "maximum'. It obviously so subtle that it went un-noticed. But I mustn't grumble, we great writers are rarely fully understood - I even confuse myself on occasions!

I mentioned Bill Jackson earlier and should have acknowledged in the article that the idea for it stemmed from assistance given to me by Bill on another RAMDISK matter. His efforts gave me a much better understanding of SPARTADOS and enabled me to put my idea to use. Yet again, thanks Bill.

John Bunting.

JUST A BYTE OF HUMOR!

Meddyical Disctionary daffynitions - for DUCTERS.

ARTERY- The study of objects d'art
BACTERIA- Back door of a Cafeteria
BARIUM- What Doctor's do when treatment fails
BOWEL- A letter like A,E,I,O,U
CESAREAN SECTION- A district in ROME
CATSCAN- Searching for the Kitty- Cat
CAUTERIZE-Made eye contact with her
COLIC- A sheep dog
COMA- A punctuation mark
CONGENITAL-A friendly person
D&C- Where WASHINGTON is
DILATE- To live too long
ENEMA- Not a friend at all
FESTER- Quicker

A BYTE OF HUMOR continued

FIBULA- A small lie
GEITLA- A non-Jewish person
G.I.SERIES-A soldiers ballgame
GRIPPE- A suitcase
HANGNAIL- A coat hook
HIGH COLONIC-Jewish-religious Holiday
IMPOTENT- A distinguished, well known person

Thats enough for now! Look for more in future newsletters.

The above article was taken from the OL Hacker's newsletter for Sept/Oct/93. This article really appealed to my sense of humour and I hope will appeal to yours. All at TWAUG give their thanks to the OL Hackers for their support.

NEWS FROM GERMANY.

Hello!

My name is Robert Kern, I come from the South-West of Germany, I will try to inform you about news and products from Germany.

NOW LETS START:

MAGAZINES: There are a lot of magazines, I think you will know some, eg. Zong, Atari-Magazin (PPP), Top-Mag (PD Top-Crew), Etc. Power Per Post (PPP) brought a new disk-magazine on the German market, called GYZYGY, it contains not only reports about the 8-bit Atari but also about other subjects, hopefully I'll be able to tell you about it in the next issue.

HARDWARE: A speedy for the XF551 was introduced on our market. It should be compatible on the Speedy 1050, sorry I can't tell you more, but it costs DM 179,-.

PRINTER PROGRAMS: Sprint XL (ABBUC) is a print processor, like Signum on Atari ST. It works with nearly all word processing programs, like Textpro, Atari-Writer, Startexter, Speedscript, etc. On this disk are an additional 39 fonts. The printouts are of high quality, one of the best programs available. The price is DM 30,-.

GAMES: Like GB, Germany is flooded with software (games) from Poland, but these are not the only high quality games. In Germany a game appeared called Mister X. Yes, this is a conversion of the parlour game Scotland Yard. I can't tell you anything about the quality of this game not having tried it yet. The best games on the German market are Herbert part I & II, your job is to lead your duck home safely, very good graphics and nice sound.

I finish this overview about the products from Germany by saying, if you are interested in any of these items, send your orders to TWAUG, Dave will collect the orders and send them to me, I will provide these articles at a good price.

Bye for now. Robby.

CRACKING THE CODE

by Keith Mayhew

Re-printed by M. Gerum

This article first appeared in "The UK ATARI Computer Owners Club" later renamed "MONITOR"

PART 8

We finished last time with a program which accessed a graphics 8 screen, randomly setting pixels on or off. This time we will start with a description of a more advanced program which allows drawing on a graphics 8 screen with a joystick.

THE PROGRAM

The assembly language for the drawing program is shown in Listing 1, the assembled code can be loaded by the BASIC program shown in Listing 2. Assuming the code has now been loaded, it can be run by typing the following line into BASIC.

```
GRAPHICS24:X=USR(20480)
```

Before we start looking at the program in detail, here is a short description of what the program actually does. A cursor is displayed on the screen in the form of a cross-hair which can be moved around with a joystick. If the joystick button is pressed then points will be plotted on the screen under the cross. There are four keys on the keyboard which are looked at, they are the "E" key which causes any further drawing to erase when the button is pressed, the "D" key will switch back to normal drawing mode. The keys "S" and "F" will change the speed of drawing from slow to fast respectively; initially slow is selected. Lastly, if one of the START, SELECT or OPTION keys are pressed then the program will return back to BASIC.

Now we know what the program does we can start looking at the assembly language listing. The first five equates define system location. The new locations here are "STICK" which holds the up/down/left/right information for joystick 0, (STRIG) which holds the state of the button for joystick 0, and CH which holds a code for the most recent key pressed, next is a list of the program's variables starting at CB hex. PLTMOD holds a value which determines how a point is plotted, if it is 0 then the point will be erased, if it is 1 then the point will be drawn and if it is anything else it will cause the point to be inverted. The plotting mode values are defined from line 300 to 320 as ERA, DRW and INV respectively.

COLOUR is used to hold the current plotting mode used for drawing with the joystick and is either "erase" or "draw". XPOS and YPOS hold the current pixel position, note that the X position is a two byte variable because there are 320 pixels across a graphics 8 screen.

The maximum values for X and Y are defined as XMAX and YMAX on lines 260 and 270. CURSON is set to FF hex when the cursor is displayed and to zero when it is turned "off". DELVAL holds the delay value which determines how fast the cursor can move around the screen, the two speeds used are defined as SLOW and FAST on lines 280 and 290. The last few variables are used within subroutines only and their use will be seen later.

The program starts initialising the X and Y positions to half their maximium values. The colour is set to "draw" and the speed to "slow". A call to cursor draws the cross-hair in the initial position and then the CURSON flag is set to FF hex to indicate the cursor is on. This value is also stored in CH which clears any key code which might have been stored there.

The main loop starts by looking at the state of the consol keys, if any are down then the program exits. If not, then the key code is loaded from CH and compared against the value of 3A, 2A, 3E and 38 hex which are the codes for the upper case letters D, E, S and F respectively, they are not ASCII values for those letters but an awkward internal code for the keyboard. If any of these keys are detected then the appropriate value is stored in COLOUR or DEVAL. If one of the keys has been pressed then FF hex is stored back afterwards so that the key is not detected the next time around the loop.

The value of STICK0 is examined next, if it equal to 0F hex, then the joystick has not been moved from its center position. If it had been moved then the cursor is turned off by calling the cursor routine again and the move routine is called to move the cursor in the appropriate direction. The button state of the joystick is then examined, if it is up it has the value of one and zero if it is pressed. If the button is down then the cursor is turned off if it is currently on; this would happen if the cursor had not been moved. The plotting mode is then set to the current colour value and the point is plotted. Lastly, if the cursor is on then neither the joystick has been moved or the button pressed and a jump is made back to the start of the loop.

If one of the consol keys has been pressed then the program jumps to "EXIT" which resets any key code which might have been generated and returns back to BASIC. We will now finish with a study of the four subroutines called by the main program, they are: MOVE, CURSOR, PLOT and DELAY.

Listing 1

```
100 SAVMSC  =     $58      ;Screen pointer.
110 STICK0  =     $0278    ;Joystick 0 value.
120 STRIG0  =     $0284    ;Joystick 0 button value.
130 CH      =     $02FC    ;Key press value.
140 CONSOL  =     $D01F    ;Consol keys.
150         *=    $CB      ;Set to page zero area.
160 PZERO   *=    *+2      ;Pointer to access screen.
170 TMPLOT  *=    *+2      ;Temporary for screen access.
180 PLTMOD  *=    *+1      ;Plotting mode.
190 COLOUR  *=    *+1      ;Colour value.
200 XPOS    *=    *+2      ;Current X position.
210 YPOS    *=    *+1      ;Current Y position.
220 CURSON  *=    *+1      ;Cursor on/off flag.
230 TMPCRS  *=    *+1      ;Temporary to hold X or Y.
240 DELVAL  *=    *+1      ;Delay value.
250 COUNT   *=    *+1      ;Count value.
260 XMAX    =     319      ;Maximum X pixel number.
270 YMAX    =     191      ;Maximum Y pixel number.
280 SLOW    =     $10      ;Slow speed delay value.
290 FAST    =     1        ;Fast speed delay value.
300 ERA     =     0        ;Erase colour.
```

CRACKING THE CODE continued

```
0310 DRW     =     1        ;Drawing colour.
0320 INV     =     $FF      ;Inverse mode.
0330         *=    $5000    ;Start program at 20K.
0340         PLA            ;Clean up stack.
0350         LDA   #XMAX/2  ;Load half of maximum X value.
0360         STA   XPOS     ;Save as start X position.
0370         LDA   #0       ;Set high byte of
0380         STA   XPOS+1   ; X position to zero.
0390         LDA   #YMAX/2  ;Load half of maximum Y value.
0400         STA   YPOS     ;Save as start Y position.
0410         LDA   #DRW     ;Set initial colour
0420         STA   COLOUR   ; to 'draw'.
0430         LDA   #SLOW    ;Set initial delay
0440         STA   DELVAL   ; to 'slow'.
0450         JSR   CURSOR   ;Draw the cursor.
0460         LDA   #$FF     ;Set cursor flag
0470         STA   CURSON   ; to 'on'.
0480         STA   CH       ;Reset key code value.
0490 LOOP    LDA   CONSOL   ;Get consol keys.
0500         CMP   #7       ;If any are down
0510         BNE   EXIT     ; then exit.
0520         LDA   CH       ;Get key code.
0530         CMP   #$FF     ;If no key
0540         BEQ   TSTICK   ; then skip key tests.
0550         CMP   #$3A     ;See if it is code for 'D'.
0560         BNE   TSTERA   ;No, then try next.
0570         LDA   #DRW     ;Set colour
0580         STA   COLOUR   ; to 'draw'.
0590         JMP   RSETCH   ;Skip rest of key tests.
0600 TSTERA  CMP   #$2A     ;See if 'E'.
0610         BNE   TSTSLW   ;No, then try next.
0620         LDA   #ERA     ;Set colour
0630         STA   COLOUR   ; to 'erase'.
0640         JMP   RSETCH   ;Skip rest.
0650 TSTSLW  CMP   #$3E     ;See if 'S'.
0660         BNE   TSTFST   ;No, then try next.
0670         LDA   #SLOW    ;Set speed
0680         STA   DELVAL   ; to 'slow'.
0690         JMP   RSETCH   ;Skip last test.
0700 TSTFST  CMP   #$3B     ;See if 'F'.
0710         BNE   RSETCH   ;Skip if not.
0720         LDA   #FAST    ;Set speed
0730         STA   DELVAL   ; to fast.
0740 RSETCH  LDA   #$FF     ;Reset key code.
0750         STA   CH
0760 TSTICK  LDA   STICK0   ;Get stick value.
0770         CMP   #$0F     ;If no movement
0780         BEQ   TSTBUT   ; then skip.
0790         JSR   CURSOR   ;Else turn off cursor.
0800         JSR   MOVE     ;Update X and Y position.
0810 TSTBUT  LDA   STRIG0   ;Get button value.
0820         BNE   DRWCRS   ;Skip if up.
0830         LDA   CURSON   ;Else see if cursor is off.
0840         BEQ   PLTPNT   ; Yes, then skip.
0850         JSR   CURSOR   ;Else turn it off.
0860 PLTPNT  LDA   COLOUR   ;Get current colour.
0870         STA   PLTMOD   ;Save as plotting mode.
0880         JSR   PLOT     ;Plot the current point.
0890 DRWCRS  LDA   CURSON   ;See if cursor is on.
0900         BNE   LOOP     ;Yes then go to start.
0910         JSR   CURSOR   ;Else turn cursor back on.
0920         JSR   DELAY    ;Produce a delay.
0930         JMP   LOOP     ;Go to start.
0940 EXIT    LDA   #$FF     ;Clear any
0950         STA   CH       ; key code.
0960         RTS            ;Return.
0970 ;
0980 ;Move X and Y depending on stick value.
0990 MOVE    LDA   STICK0   ;Get stick value.
1000         AND   #1       ;Look at 'up' bit.
1010         BNE   DOWN     ;Skip if not down.
1020         LDA   YPOS     ;Get Y value.
1030         BEQ   LEFT     ;Skip if at zero.
1040         DEC   YPOS     ;Else decrement Y.
1050         JMP   LEFT     ;Go to 'left' test.
1060 DOWN    LDA   STICK0   ;Get stick
1070         AND   #2       ; 'down' bit.
1080         BNE   LEFT     ;If not down then skip.
1090         LDA   YPOS     ;Else get Y value.
1100         CMP   #YMAX    ;If at maximum
1110         BEQ   LEFT     ; then skip.
1120         INC   YPOS     ;Else increment Y value.
1130 LEFT    LDA   STICK0   ;Get stick
1140         AND   #4       ; 'left' bit.
1150         BNE   RIGHT    ;Skip if not down.
1160         LDA   XPOS     ;Get X value.
1170         BNE   LEFT2    ;If not zero then OK.
1180         LDA   XPOS+1   ;Else if high byte is
1190         BEQ   MOVEXT   ; zero then skip rest.
1200         DEC   XPOS+1   ;Else decrement high byte as well.
1210 LEFT2   DEC   XPOS     ;Decrement low byte of X.
1220         JMP   MOVEXT   ;Skip 'right' test.
1230 RIGHT   LDA   STICK0   ;Get stick
1240         AND   #8       ; 'right' value.
1250         BNE   MOVEXT   ;Skip if not down.
1260         LDA   XPOS+1   ;Get X high byte.
1270         BEQ   RIGHT1   ;If zero OK.
1280         LDA   XPOS     ;Else if X low byte
1290         CMP   #XMAX1&$FF ; is at maximum then
1300         BEQ   MOVEXT   ; skip rest.
1310 RIGHT1  INC   XPOS     ;Increment low byte.
1320         BNE   MOVEXT   ;Skip if not zero.
1330         INC   XPOS+1   ;Else ajust high byte.
1340 MOVEXT  RTS            ;Return
1350 ;
1360 ;Draw cursor in 'inverse' mode.
1370 CURSOR  LDA   CURSON   ;Get cursor state.
1380         EOR   #$FF     ;Flip state.
1390         STA   CURSON   ;Save back.
1400         LDA   #INV     ;Set plot mode
1410         STA   PLTMOD   ; to 'inverse'.
```

CRACKING THE CODE continued

```
1420          JSR   PLOT    ;Plot current point.
1430          LDA   YPOS    ;Save Y value.
1440          STA   TMPCRS
1450          LDA   #4      ;Set count to 4 pixels.
1460          STA   COUNT
1470 CRSUP1   LDA   YPOS    ;Get Y value.
1480          BEQ   CRSDN   ;If zero then skip.
1490          DEC   YPOS    ;Else decrement it.
1500          JSR   PLOT    ;Plot point.
1510          DEC   COUNT   ;Decrement count
1520          BNE   CRSUP1  ;Continue if more.
1530 CRSDN    LDA   TMPCRS  ;Restore Y value.
1540          STA   YPOS
1550          LDA   #4      ;Reset count.
1560          STA   COUNT
1570 CRSDN1   LDA   YPOS    ;Test Y value
1580          CMP   #YMAX   ; against maximum.
1590          BEQ   CRSLT   ;Skip if equal.
1600          INC   YPOS    ;Else increment it.
1610          JSR   PLOT    ;Plot point.
1620          DEC   COUNT   ;Decrement count.
1630          BNE   CRSDN1  ;Continue if more.
1640 CRSLT    LDA   TMPCRS  ;Restore Y value.
1650          STA   YPOS
1660          LDA   XPOS    ;Save X value.
1670          STA   TMPCRS
1680          LDA   XPOS+1  ;Save X high byte
1690          PHA           ; on the stack.
1700          LDA   #4      ;Reset count.
1710          STA   COUNT
1720 CRSLT1   LDA   XPOS    ;See if X is zero.
1730          BNE   CRSLT2  ;OK if not.
1740          LDA   XPOS+1  ;Else if high byte
1750          BEQ   CRSRT   ; zero then skip.
1760          DEC   XPOS+1  ;Else decrement high byte.
1770 CRSLT2   DEC   XPOS    ;Decrement low byte.
1780          JSR   PLOT    ;Plot point.
1790          DEC   COUNT   ;Decrement count.
1800          BNE   CRSLT1  ;Continue if more.
1810 CRSRT    LDA   TMPCRS  ;Restore X value.
1820          STA   XPOS
1830          PLA           ;Get high byte and save
1840          PHA           ; another copy on the stack.
1850          STA   XPOS+1  ;Restore high byte.
1860          LDA   #4      ;Reset count.
1870          STA   COUNT
1880 CRSRT1   LDA   XPOS+1  ;Get high byte.
1890          BEQ   CRSRT2  ;OK if zero.
1900          LDA   XPOS    ;Else test low byte against
1910          CMP   #XMAX&$FF ; maximum value.
1920          BEQ   CRSEXT  ;If equal then skip.
1930 CRSRT2   INC   XPOS    ;Else increment low byte.
1940          BNE   CRSRT3  ;Skip if not zero.
1950          INC   XPOS+1  ;Increment high byte.
1960 CRSRT3   JSR   PLOT    ;Plot point.
1970          DEC   COUNT   ;Decrement count.
1980          BNE   CRSRT1  ;Continue if more.
1990 CRSEXT   LDA   TMPCRS  ;Restore X low byte.
2000          STA   XPOS
2010          PLA           ;Restore X high byte.
2020          STA   XPOS+1
2030          RTS           ;Return.
2040 ;
2050 ;Plot a point at current X and Y.
2060 PLOT     LDA   SAVMSC  ;Copy screen pointer
2070          STA   PZERO   ; into our pointer.
2080          LDA   SAVMSC+1
2090          STA   PZERO+1
2100          LDA   YPOS    ;Get Y vaue.
2110          STA   TMPLOT  ;Save in temporary location.
2120          LDA   #0      ;Set high byte to zero.
2130          STA   TMPLOT+1
2140          LDX   #3      ;Set for eight times...
2150 MULT8    ASL   TMPLOT  ;Shift left on both bytes.
2160          ROL   TMPLOT+1
2170          DEX           ;Decrement count
2180          BNE   MULT8   ;Loop.
2190          LDA   PZERO   ;Get pointer value.
2200          CLC
2210          ADC   TMPLOT  ;Add low byte.
2220          STA   PZERO   ;Save back.
2230          LDA   PZERO+1 ;Same for high byte.
2240          ADC   TMPLOT+1
2250          STA   PZERO+1
2260          ASL   TMPLOT  ;Shift twice more
2270          ROL   TMPLOT+1 ; to get 32 times.
2280          ASL   TMPLOT
2290          ROL   TMPLOT+1
2300          LDA   PZERO   ;Add to pointer value.
2310          CLC
2320          ADC   TMPLOT
2330          STA   PZERO
2340          LDA   PZERO+1 ;And high byte...
2350          ADC   TMPLOT+1
2360          STA   PZERO+1
2370          LDA   XPOS+1  ;Get high byte of X.
2380          LSR   A       ;Shift low bit into carry.
2390          LDA   XPOS    ;Get low byte.
2400          ROR   A       ;Rotate carry in.
2410          LSR   A       ;Shift twice more to
2420          LSR   A       ; divide X by a total of 8.
2430          TAY           ;Save in Y register as index.
2440          LDA   XPOS    ;Get low byte of X.
2450          AND   #7      ;Mask to get low three bits.
2460          TAX           ;Move to X register.
2470          LDA   #$80    ;Load with high bit set.
2480 SHIFT    DEX           ;Decrement count.
2490          BMI   PLOT1   ;Skip if shifting finished.
2500          LSR   A       ;Else shift.
2510          JMP   SHIFT   ;Continue if more.
2520 PLOT1    LDX   PLTMOD  ;Get plotting mode.
2530          BEQ   ERASE   ;If zero then 'erase' mode.
```

CRACKING THE CODE continued

```
2540          CPX     #DRW     ;If 'draw' mode then
2550          BEQ     DRAW     ; skip to it.
2560          EOR     (PZERO),Y ;'Exclusive-or' for 'inverse'.
2570          JMP     PLTEXT   ;Skip rest.
2580 DRAW     ORA     (PZERO),Y ; 'Or' for 'draw' mode.
2590          JMP     PLTEXT   ;Skip rest.
2600 ERASE    EOR     #$FF     ;Invert bits.
2610          AND     (PZERO),Y ; and 'and' for 'erase'.
2620 PLTEXT   STA     (PZERO),Y ;Save value back to screen.
2630          RTS              ;Return.
2640 ;
2650 ;Produce a small delay.
2660 DELAY    LDX     DELVAL   ;Get delay value.
2670 DELAY1   LDY     #$FF     ;Inner delay loop.
2680 DELAY2   DEY
2690          BNE     DELAY2   ;Loop.
2700          DEX              ;Decrement outer loop value.
2710          BNE     DELAY1   ;Loop.
2720          RTS              ;Return.
```

Listing 2

```
QZ 10 DIM HEX$(16)
TU 20 LINE=10000:TRAP 100:J=0:START=20480
VA 30 READ HEX$,CHKSUM:SUM=0
AA 40 FOR I=1 TO 15 STEP 2
ZG 50 D1=ASC(HEX$(I,I))-48:D2=ASC(HEX$(I+
   1,I+1))-48
KT 60 NUM=((D1-7*(D1>16))*16+(D2-7*(D2>16
   )))
LW 70 SUM=SUM+NUM:POKE START+J,NUM:J=J+1:
   NEXT I
LY 80 IF SUM=CHKSUM THEN LINE=LINE+10:GOT
   O 30
IN 90 ? "Checksum error on this line:"
VO 95 LIST LINE:END
YS 100 PRINT "Data in memory."
QM 10000 DATA 68A99185D1A90085,1076
JG 10010 DATA D2A9558 5D3A901B5,1121
ET 10020 DATA D0A91985D620DB50,1071
WE 10030 DATA A9FF85D48DFC02AD,1337
KD 10040 DATA 1FD0C987D062ADFC,1178
TN 10050 DATA 02C9FFF02EC93AD0,1211
KX 10060 DATA 07A90185D04C5650,760
MH 10070 DATA C92AD007A900085D0,968
EH 10080 DATA 4C5650C93ED007A9,889
BE 10090 DATA 1085D64C5650C938,862
NY 10100 DATA D004A90185D6A9FF,1153
IX 10110 DATA 8DFC02AD7802C90F,906
KO 10120 DATA F00620DB50208E50,831
GJ 10130 DATA AD8402D00EA5D4F0,1146
FC 10140 DATA 0320DB50A5D085CF,1047
WF 10150 DATA 206151A5D4D0A020,987
FG 10160 DATA DB5020CC514C1F50,803
BB 10170 DATA A9FF8DFC0260AD78,1208
IM 10180 DATA 02290100 9A5D3F0,877
MG 10190 DATA 14C6D34CAD50AD78,1051
KD 10200 DATA 022902D008A5D3C9,838
KB 10210 DATA BFF002E6D3AD7802,1169
IS 10220 DATA 2904D00FA5D1D006,856
TP 10230 DATA A5D2F01EC6D2C6D1,1460
DT 10240 DATA 4CDA50AD7802290 8,718
QY 10250 DATA D010A5D2F006A5D1,1219
WZ 10260 DATA C93FF006E6D1D002,1159
TE 10270 DATA E6D260A5D449FF85,1374
BQ 10280 DATA D4A9FF85CF206151,1186
XB 10290 DATA A5D385D5A904 85D7,1243
CA 10300 DATA A5D3F009C6D32061,1163
NC 10310 DATA 51C6D7D0F3A5D585,1456
OK 10320 DATA D3A904 85D7A5D3C9,1309
ZD 10330 DATA BFF009E6D3206151,1091
WP 10340 DATA C6D7D0F1A5D585D3,1584
JG 10350 DATA A5D185D5A5D248A9,1336
JY 10360 DATA 0485D7A5D1D006A5,1105
YS 10370 DATA D2F00BC6D2C6D120,1308
WT 10380 DATA 6151C6D7D0EDA5D5,1414
BM 10390 DATA 85D16848B5D2A904,1034
YZ 10400 DATA 85D7A5D2F006A5D1,1343
EE 10410 DATA C93FF00DE6D1D002,1166
NK 10420 DATA E6D220 6151C6D7D0,1271
RY 10430 DATA E9A5D585D16885D2,1400
RK 10440 DATA 60A55885CBA55985,1072
YU 10450 DATA CCA5D385CDA900 85,1220
IM 10460 DATA CEA2030 6CD26CECA,1020
DA 10470 DATA D0F9A5CB1865CD85,1288
CE 10480 DATA CBA5CC65CE85CC06,1222
ER 10490 DATA CD26CE06CD26CEA5,1069
WE 10500 DATA CB1865CD85CBA5CC,1238
GN 10510 DATA 65CE85CCA5D24AA5,1258
GM 10520 DATA D16A4A4AA8A5D129,1046
DR 10530 DATA 07AAA908CA30044A,882
CF 10540 DATA 4CAC51A6CFF00EE0,1180
VA 10550 DATA 01F00551CB4CC951,888
FO 10560 DATA 11CB4CC95149FF31,955
LU 10570 DATA CB91CB60A6D6A0FF,1442
PT 10580 DATA 88D0FDCAD0F860,1351
```

The subroutine "MOVE" consists of four similar sections which load the value of the joystick and adjust the X and Y values according to the direction specified. The coding scheme used for the variable STICK0 is as follows. The upper four bits are unused and are always zero, the lower four bits are normally all ones which refers to no movement thence the value of 0F hex was used for the comparison on line 770).

These lower four bits, starting from bit 0, represent the following directions on the joystick if they are set to zero: UP, DOWN, LEFT and RIGHT. Only one or two of these bits can be set to zero at any time depending on if the joystick is at a diagonal or is pointed straight in one direction.

The direction "up" causes the Y value to be decremented unless it is already at zero, i.e the top of the screen, and the "down" test is skipped because both bits could not be set at once. The "down" direction increments the Y value unless it is already at the value of YMAX, i.e. at the bottom of the screen. The left and right directions are handled in the same way except that the X direction has a high byte which needs to be tested, but note that this byte can only take on the value of 0 or 1 depending on if the current X value is below 256 or above 255 respectively.

The "CURSOR"routine draws the cross-hair, it consists of the current pixel and four adjacent pixels in all four directions. The value in "CURSON" is complimented so that if the cursor was on it is now of and vice versa.

CRACKING THE CODE continued

The plotting mode is set to "inverse" which causes the value of any pixel plotted to be complimented . By using the inverse mode, if the cursor is drawn once it appears on the screen and if it is drawn again it will cause it to flip back to its previous state so that it cannot be seen. This also allows the cursor to be plotted over existing pixels without having to save their values before the cursor is plotted. Drawing the cursor consists of four similar sections, these draw four pixels in the four directions from the current location, before this is done the current pixel is complimented by a call to "PLOT" on line 1420. The first section of code plots four pixels in the upward direction, before each pixel is plotted the Y value is tested to see if it is at the top of the screen, in which case no more pixels are drawn in that direction. The next section draws four pixels downwards, but first the origional value of Y is restored from the temporary location "TMPCRS". The cursor left and right code is similar to the up and down code except that the high byte of the X position is also has to be saved and restored, this is achieved by using the stack.

The "PLOT" routine is fundamentallly the same as the routine used in Listing 1 last time. The Y value is loaded into the temporary locations TMPLOT and TMPLOT+1. This is then multiplied by 8 and added onto the value in PZERO and PZERO+1, which pointed to the start of the screen. The temporary value is then multiplied by another 4 to give 32 times and added onto the pointer so that, in effect, 40 times the Y value has been added. The next task is to find the offset into the line. As there are eight pixels to each byte, we need to divide the X value by eight, this is done on line 2370 to 2420. Note that only one shift is necessary on the high byte of X because only bit 0 is used. This byte offset is then saved in the Y register for indexing. We now have to find the appropriate bit which we wish to access within that byte. This is done by taking the low byte of the X value and masking off all but the lowest three bits. This number will now represent the bit number to be accessed (if we treat the left most bit of the byte as zero). This value is moved into the X register as a count, and the accumulator is loaded with 80hex so that the left most bit is set. The accumulator is then shifted to the right by the number of times specified in the X register. The value of PLTMOD then determines how we place this byte onto the screen. If it is equal to "DRW" then it is simply ORed to the screen byte and saved back, thus setting the pixel to on. To erase the pixel, PLTMOD has the value "ERA" (zero) and the byte in the accumulator is exclusive-ORed with FFhex inverting every bit so that the bit to be erased is now set to zero. This is ANDed with the screen byte setting the appropriate bit to 0 and leaving the rest unchanged. Lastly, if we are in the "inverted" mode then the byte is exclusive-ORed with the screen byte, thus inverting that pixel. Whichever method was used, the accumulator would hold the correct byte to be saved back to the screen at line 2620. The last routine is"DELAY". This uses the X and Y registers to produce a variable delay depending on the value in the variable "DELVAL". The inner loop is repeated 256 times with the Y register and this is repeated with the outer loop by the number of times specified in DELVAL.

If you can follow how the program workd then you are probably rady to write your own programs. However if you feel significantly lost with this program then it is time to start a little revision!

As promised last time, we will now start our detailed tour of the Atari hardware and operating system. Most programs will now be written to illustrate how to use a certain part of the machine. You may find a few useful techniques which you could also apply in your own code in the process.

THE MEMORY MAP

As you will be aware, there are 64K bytes of memory which the 6502 processor can address. The divirsion of this memory between the various parts of the system is usually illustrated by a "memory map". Figure 1 shows the memory map for all the 8-bit ATARI computers, namly the 400, 800, 600XL, 800XL and 130XE models. All of these machines are fundamentally the same, most differences will be noted in due course.

The memory map shown gives the sizes of the main areas on the right and to the left are the addresses of these areas, some of which are shown in "K"s for reference. The operating system occupies the top 8K of memory from E000 to FFFF hex and the 4K area from C000 to CFFF hex. Note that the latter 4K is only present in the XL and XE range. The operating system (O.S.) is, obviously, held in ROM so that it is always present, remember that if you try to write to ROM nothing will happen; you cannot change its contents. The reason why the O.S. occupies the top locations of memory is that, if you recall the top six bytes of all the address range are special to the 6502. Location FFFC and FFFD hex hold the address of the routine which will be executed on power up. Location FFFA and FFFB hex hold the addresses of the non-maskable interrupt (NMI) routine. Locations FFFE and FFFF hex hold the address of the interrupt request (IRQ) routine. The three addresses will all point to somewhere inside the O.S. ROM area. The 2K area from D800 to DFFF hex is another area of ROM which contains the floating point (F.P.) software to handle arithmetic on such numbers. The F.P. software can be considered as part of the operating system, however, it is functionally completely separate.

The two 8K regions from 8000 to BFFF hex are where the ROM cartridges reside. The idea of two seperate cartridges, "A" and "B", comes from the old 800 model and was done mainly because it was not possible to get 16K into one physical cartridge at the time. If an 8K cartridge is installed in a machine then it will occupy the "A" cartridge area, if it is 16K then it will occupy both areas "A" and "B". Note that ATARI BASIC is 8K in size and is installed as a cartridge automatically by the XL and XE machines as it is built inside the machine, it is deselected (effectively removed) by either inserting another cartridge to take its place or by holding down the OPTION key.

The size of the Ram area is different between machines but it is always present from location zero and extends upwards. Assuming you have more than 32K of memory, then if the cartridge areas are not used then the RAM will be accessable in those areas, otherwise it will retain its data but will not be accessible. The same principle applies to the XL and XE machines where there is hidden RAM in the top 16K of the machine which can be accessible.

THE HARDWARE

We will finish this time by starting an examination of the 2K area from D000 to D7FF hex which contains the hardware.

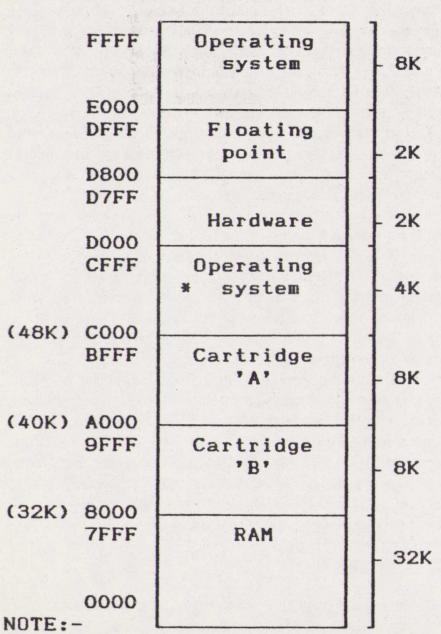
CRACKING THE CODE continued

We have already used two of these locations in our programs, namely CONSOL and RANDOM. When accessing these 'memory' locations you are actually accessing one of the special chips of the ATARI.

The areas used for the four chips are as follows. The "GITA" chip occupies the locations D000 to D0FF hex, the "POKEY" chip occupies locations D200 to D2FF hex, the "PIA" chip occupies the locations D300 to D3FF hex and the "ANTIC" chip from D400 to D4FF hex. Very few of these locations for each chip actually have a use. The hardware locations on these chips are usually called registers but the way these registers can be accessed varies from register to register, the various types are as follows. Some registers are read/write, that is they behave like RAM locations; others are read only, that is they act like ROM and others are write only, that is you cannot read back what is held in a register. The read only and write only registers often occupy the same location, that is on reading you will access one register and on writing you will access the other. Lastly there are some registers which are activated simply by writing to them, i.e. the actual data written is irrelevant.

We will continue next time with our study of the hardware. In the meantime, if you have not got a decent reference manual for the hardware and the operating system then it is highly recommended that you get ATARI's TECHNICAL REFERENCE NOTES which also contains a complete listing of the 400/800 operating system as well as the circuit diagrams. These notes can be a little heavy going, but they are very thorough and useful. A more practical guide, again recommended is COMPUTES!'s MAPPING THE ATARI (XL/XE edition).

FIGURE 1



NOTE:-
* Not present in 400/800 models

BRITISH SPARTADOS TIME

by Terry Chamberlain

Get the SpartaDOS Real Time Clock to Keep Proper Time - and Fix the Date Notation into the Bargain

THE PROBLEM

As an otherwise firm convert to the advantages of SpartaDOS over the other varieties of DOS available on the 8-bit Atari, I have always been somewhat annoyed that the internal real-time clock runs slow - losing 10 minutes an hour in normal use. This criticism was also made by John S. Davison in his review of SpartaDOS back in Issue 32 of Page 6 magazine and, as he suggested, the problem is due to the built-in clock being designed to run in the USA with a 60Hz VBI. Operating in the UK, or anywhere else, with a PAL system and 50Hz VBI makes the clock 'tick' that much more slowly and hence lose time.

For the rich amongst us an instant solution is to purchase the ICD R-Time 8 cartridge which runs its own internal hardware clock independently of the VBI. However, for users of SpartaDOS Version 3.2d on the XL/XE machines, there is a much cheaper (and easy) fix. Read on to save yourself some money ...

THE METHOD

Let's start with a little background information. The SpartaDOS time-of-day clock uses the standard Atari real-time clock register RTCLOK at locations 18, 19, and 20 (hex $12 to $14). These are incremented during every Immediate VBI, ie. 50 times a second, by the standard Atari Operating System. The accumulated count is read periodically by a SpartaDOS routine which uses it to update SpartaDOS's own internal clock and calendar registers (TIMER and DATER - see the SpartaDOS Construction Set Manual p.110). The latter are then used when putting time/date stamps on files, and by the TDLINE utility in displaying current time and date at the top of the screen.

The SpartaDOS update routine uses four built-in constants to convert the 'jiffy' count in RTCLOK to seconds, minutes, hours, and days. These are, as you might expect, based on the USA standard of 60 jiffies per second, ie. -

```
$00003C =      60 jiffies/sec
$000E10 =    3600 jiffies/min
$034BC0 =  216000 jiffies/hr
$4F1A00 = 5184000 jiffies/day
```

To correct the timekeeping for British and other PAL systems running at 50 jiffies per second we need a different set of constants as follows -

```
$000032 =      50 jiffies/sec
$000BB8 =    3000 jiffies/min
$02BF20 =  180000 jiffies/hr
$41EB00 = 4320000 jiffies/day
```

- and some means of inserting these numbers into the SpartaDOS code.

BRITISH SPARTADOS TIME continued

THE SOLUTION

To make things difficult the relevant routine is neatly tucked away in the RAM underlying the OS ROM. My initial attempts to patch this code in situ, after loading SpartaDOS, were not very successful - usually causing a lock-up or crash.

The solution finally adopted is much simpler and works by patching the SpartaDOS X32D.DOS file directly on disc. The amended DOS file can then be used to boot the computer as normal - but now with a clock running at the correct speed.

When you use TDLINE with this fix you will notice that the seconds on the displayed time increment a little jerkily. TDLINE is intended to update its display line every half-second, which it does by counting 30 VBIs. For UK use this count needs to be changed to 25 by altering a single byte in the TDLINE.COM file from $1E to $19.

While we are busy patching the time-keeping parameters it is also quite easy to dispose of the odd American custom of handling dates in a Month-Day-Year format, and change to our more familiar form of Day-Month-Year. What makes this particular fix slightly absurd is that SpartaDOS actually stores the date internally (and in its on-disc time/date stamps) in DD-MM-YY format already. The input and output routines which handle the date swap the DD and MM values over specially to suit our American cousins. We can 'unswap' the date values by interchanging two pairs of bytes in X32D.DOS - the bytes are part of temporary memory addresses used by SpartaDOS in its date manipulations.

THE PROGRAM

The short BASIC program shown performs all of the necessary patches for you. It will patch any SpartaDOS disc containing X32D.DOS and TDLINE.COM. For those interested, the SpartaDOS DUMP command can be used to examine the contents of X32D.DOS and TDLINE.COM before and after the operation.

The revised constant values are patched into the X32D.DOS file, at file address $2538, and the swap of byte values to change the date format occurs between file addresses $1B8D/$1B9B for output, and $3060/$3066 for input. The single patch to TDLINE.COM is made at file address $036D.

CAUTION : Do NOT attempt this patch operation on your original SpartaDOS Master disc - or even your usual working copy. Make a further copy and experiment using this until you are sure that everything is working properly.

YOU HAVE BEEN WARNED !

Remember to UNLOCK the copy disc and UNPROTECT the files to be patched before running the program - then follow the prompt.

After patching X32D.DOS and TDLINE.COM the only change in using SpartaDOS involves the DATE command. You will see that the current date is displayed in DD-MM-YY format, and you should enter the new date in the same way. When you display a directory of any SpartaDOS disc you will also see that the time/date stamps are shown in DD-MM-YY format.

A FINAL NOTE

Because the internal time-of-day clock is driven by the VBI, any program which disables the VBI, or interferes with the Immediate VBI vector, will stop the clock. Inhibiting Deferred VBI processing (as happens during I/O operations) only prevents update of the SpartaDOS time registers - RTCLOK keeps on counting in the background. When Deferred VBI processing resumes again the contents of RTCLOK are used to update the current time (that's what the four constants are used for).

And before anyone leaps to protest, I have to admit that the clock still runs slightly slow by about 10 seconds per hour. The accuracy is set directly by the frequency of the CPU clock oscillator fitted to the XL/XE PAL machines (3.546894MHz) and hardware counters within the ANTIC chip, and cannot be adjusted through any software fix. If you're not happy with this you'll just have to go out and buy that R-Time 8 cartridge after all ...

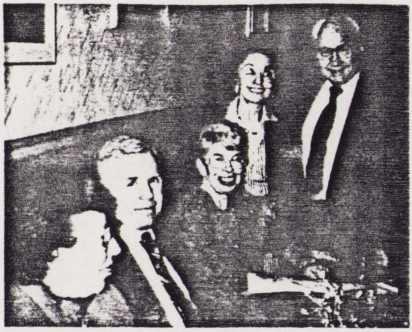
Terry Chamberlain
9 Coniston Way
Macclesfield
Cheshire, SK11 7XR

Tel. 0625 421324

```
100 REM ************************
110 REM *                      *
120 REM *  SPARTADOS TIME AND DATE *
130 REM *                      *
140 REM *       ADJUSTMENTS    *
150 REM *                      *
160 REM ************************
170 REM * Terry Chamberlain - Apr90  *
180 REM ************************
190 REM
200 Z=0:DIM D$(12),F$(15):F$(1,3)="D1:"
210 ? :? :? "Insert SpartaDOS Disc in Drive 1"
220 ? :? "  Press <RETURN> When Ready ..":?
230 CLOSE #1:OPEN #1,4,0,"K:"
240 GET #1,A:IF A()155 THEN 240
250 READ D$,A:IF D$="DONE" THEN 300
260 F$(4)=D$:? D$
270 CLOSE #1:OPEN #1,9,0,F$:POINT #1,A,Z
280 READ A:IF A()-1 THEN PUT #1,A:GOTO 280
290 GOTO 250
300 CLOSE #1:END
310 DATA X32D.DOS,9533,50,0,11,184,2,191,32,65,235,0,-1
320 DATA X32D.DOS,7053,250,-1
330 DATA X32D.DOS,7067,251,-1
340 DATA X32D.DOS,12384,74,-1
350 DATA X32D.DOS,12390,75,-1
360 DATA TDLINE.COM,877,25,-1
370 DATA DONE,-1
```

The above listing can be found on the issue disk under the name, (BSPDTIME.BAS).

THE OL'HACKERS USER GROUP



The OL Hackers are based in New York in the U.S.A who are a group of dedicated 8-bit supporters. They produce an excellent bi-monthly disk based newsletter which is full of articles, programs, etc... Some of us at TWAUG have subscribed to the OL Hackers for a couple of years and when TWAUG first started, they were the first overseas user group to subscribe to the TWAUG newsletter. We are very grateful for the help and support they have given over the last year.

As you will have noticed, we have a number of back issues in our PD library and they have also sent us a number of other programs to add to our library which are very welcome. If you haven't already seen any of their newsletters, then I suggest that you get one from our library, I'm sure you'll enjoy it.

Above, there are two photographs of some of the members of the OL Hackers and their good ladies enjoying themselves at their annual dinner. Wouldn't it be nice if we could have something like that over here?

I would like to finish by saying "keep 8-bitting", let's all keep the 8-bit alive.

DAVID

BUY and SALE SECTION

FOR SALE.

I have a few light guns for sale. These are C/64 light guns easily adapted for use with the Atari 8-bit computer. They are fitted with ajustable sites which is very useful.

The light guns are £12 each and can be supplied with full instructions for modifying it yourself, or I would be willing to do the modification for you. If you are interested, then contact:

Mr. D. Heard-White
34 Mayfield Cre.
Brighton
BN1 8HQ.

ATARI CLASSICS

The Magazine for the Dedicated 8-Bit User

ENERGIZE YOUR 8-BIT: Plug into AC! ATARI CLASSICS, that is!
SUBSCRIBE NOW !!
Rates: Foreign (All) 3rd Class Mail $32 for 1 year
Europe/Mediterranean AIRMAIL $38 for 1 year
Payment by I.O.M or VISA - MASTERCARD
(Credit Card orders $2 processing fee under $100)

ATARI CLASSICS, 179 Sproul Road/RT.352, Frazer, PA 19355-1958
or Phone 313-973-8825.

THE OL'HACKERS ATARI USER GROUP INC.

O.H.A.U.G. is an all 8-bit user group in the State of New York, they are producing a bi-monthly first class informative newsletter on disk.

The disk is double sided full of news, views, articles and bonus games and/or utilities. The disk has its own printing utility which you can use to read the content of the disk on screen or make hard copies.

A large PD Library is also available.

Some of the T.W.A.U.G. members are contributing to the OL'Hackers newsletter and the OL'Hackers are contributing to the T.W.A.U.G. missive.

For more information on how to contribute to the newsletter write to:

O.H.A.U.G.
c/o A. Pignato
3376 Ocean Harbor Drive
Oceanside, N.Y.11572
U.S.A.

ULTIFONT

ULTIFONT has been written exclusively for T.W.A.U.G... and it is only available from our library, for £4.95.

It is one the most powerful character set editor written for the Atari. It fully supports the Atari 4-colour text modes and allows you to edit two adjacent characters as one. In addition, you may edit 2 different fonts at once and easily work between them.

It is totally joystick-driven, so you can sit back and relax, rather than hunching over the keyboard. ULTIFONT requires a colour display and 48K of memory.

Printed instructions come with the single sided disk.

More Support From

RICHARD GORE

Classic US software re-releases:-

JAWBREAKER Guide your teeth around a maze chomping candy. Includes two game mazes, originally sold as two games. Double the value!

MOUSEKATTACK Plumb the levels of Rat Alley. Classic arcade action for one or two players on the screen at once.

PRICES: £4.50 each, available on disk, standard tape or Rambit Turbo Tape. Please state which.

INTRODUCTORY SPECIAL: Buy both Jawbreaker and Mousekattack for only £6.95

Software prices include p&p within the UK, overseas add £1 per order.

Still available: The YORKY 256k plug-in memory upgrade, only £50 + £2 p&p.

You may order from:

RICHARD GORE, 79 SPROTBROUGH ROAD, SPROTBROUGH, DONCASTER, DN5 8BW, ENGLAND

PHOENIX.

The new disk based news letter from Ireland, produced by Robert Paden.
Robert recently had to give up producing the I.A.U newsletter after four issues because of lack of support.

PHOENIX will be a double sided disk, side 'A' will be packed full of text files containing Articles, reviews and much much more. Side 'B' will contain a good selection of PD software.
The first 6 issues will be available from either TWAUG, or from New Atari User PD libraries at £2.50 per issue.

If PHOENIX proves to be a success (which we hope it will), then from issue 7, it will only be available from Robert Paden himself.

MICRO DISCOUNT

Now the largest Mail Order Stockist of Atari 8 Bit Hardware & Software

THE NEW FLOPPY BOARD
FOR THE 8-BIT.

Derek Fern of MICRO DISCOUNT has informed us that he expects to be releasing his new floppy board at the SAMS show for the Atari 8-bit that will allow you to connect a 3.5 drive to your system.

The cost of the new board will be between £40 and £45. There is a possibility he may have some 3.5 drives available at the show and if so, he will hopefully be selling the board complete with a drive for about £55.

Derek would like anyone who is interested in the new board to get in touch and let him know so that he can get an idea as to possible sales.

He is also hoping to be shortly announcing another board that will allow you to connect a hard drive to your 8-bit which will also be at a very reasonable price. Why not get in touch with Derek and let him know if you are interested.

MICRO DISCOUNT
265 Chester Road
STREETLY
West Midlands B74 3EA
Tel:021-353-5730 or Fax:021-352-1669

NEW ATARI USER PAGE 6

The only magazine left in this country that supports the 8-bit

There is a large Public Domain Library available. Your support is needed to keep the magazine going.

It is now only available by subscription, for more details write to:

PAGE 6
P.O.BOX 54
STAFFORD ST16 1TB